IMS
Version 11

Database Utilities

SC19-2435-05

IMS
Version 11

Database Utilities

SC19-2435-05

Note
Before using this information and the product that it supports, be sure to read the general information under "Notices" on page 441.

This edition applies to IMS Version 11 (program number 5635-A02) and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright IBM Corporation 1974, 2015.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this information vii
Prerequisite knowledge vii
IMS function names used in this information . . . vii
How new and changed information is identified . . vii
How to read syntax diagrams viii
Accessibility features for IMS Version 11 ix
How to send your comments x

Changes to the IMS library xi

Part 1. Definition and initialization utilities 1

Chapter 1. Database Prefix Resolution utility (DFSURG10) 3
Examples for the DFSURG10 utility 11

Chapter 2. Database Prefix Update utility (DFSURGP0) 13
Control statements for the DFSURGP0 utility . . . 18
Examples for the DFSURGP0 utility 19
Restarting after an abnormal termination of the DFSURGP0 utility 24

Chapter 3. Database Scan utility (DFSURGS0) 25
Control statements for the DFSURGS0 utility . . . 30
Examples for the DFSURGS0 utility 32
Restarting after an abnormal termination of the DFSURGS0 utility 33

Chapter 4. Database Surveyor utility (DFSPRSUR) 35
Control statement for the DFSPRSUR utility . . . 38
Examples for the DFSPRSUR utility 40

Chapter 5. DEDB Initialization utility (DBFUMIN0) 49
Control statement for the DBFUMIN0 utility . . . 52
Examples of the DBFUMIN0 utility 54

Chapter 6. DEDB Sequential Dependent Delete utility (DBFUMDL0) 55
Control statements for the DBFUMDL0 utility . . . 56
Examples for the DBFUMDL0 utility 60
Range of processing for the DBFUMDL0 utility . . 61

Chapter 7. DEDB Sequential Dependent Scan utility (DBFUMSC0) 63
Control statements for the DBFUMSC0 utility . . . 66
Examples for the DBFUMSC0 utility 71
Range of scans for the DBFUMSC0 utility 71

Chapter 8. HALDB Partition Data Set Initialization utility (DFSUPNT0) 75
Control statement for the DFSUPNT0 utility . . . 79
Examples for the DFSUPNT0 utility 80

Chapter 9. HALDB Partition Definition utility (%DFSHALDB). 81
Control statements for the %DFSHALDB utility . . 87
Examples for the %DFSHALDB utility 88
Running the %DFSHALDB utility 89
HALDB Partition Definition utility ISPF panels . . 89

Chapter 10. MSDB Maintenance utility (DBFDBMA0). 117
Control statements for the DBFDBMA0 utility . . 121
Examples for the DBFDBMA0 utility 124
Functions of the MSDB Maintenance utility . . . 126

Part 2. Backup utilities 129

Chapter 11. Database Image Copy utility (DFSUDMP0) 131
Control statement for the DFSUDMP0 utility . . . 139
Examples for the DFSUDMP0 utility 140

Chapter 12. Database Image Copy 2 utility (DFSUDMT0) 143
Control statements for the DFSUDMT0 utility . . 151
Running the DFSUDMT0 utility 160

Chapter 13. Online Database Image Copy utility (DFSUICP0) 169
Control statement for the DFSUICP0 utility . . . 174
Examples for the DFSUICP0 utility 175
Restarting an image copy job after a DFSUICP0 failure 176

Part 3. Recovery utilities 177

Chapter 14. Batch Backout utility (DFSBBO00) 179
Utility control statements for the DFSBBO00 utility 188
Example for the DFSBBO00 utility 192
Running the DFSBBO00 utility in an RSR environment. 193

Chapter 15. Database Change Accumulation utility (DFSUCUM0) . . 195
Control statements for the DFSUCUM0 utility . . 201

© Copyright IBM Corp. 1974, 2015

Examples for the DFSUCUM0 utility 208

Chapter 16. Database Recovery utility (DFSURDB0) 211
Control statements for the DFSURDB0 utility . . . 221
Examples for the DFSURDB0 utility 223

Chapter 17. DEDB Area Data Set Compare utility (DBFUMMH0) 229
Control statements for the DBFUMMH0 utility . . 232
Examples for the DBFUMMH0 utility 233

Chapter 18. DEDB Area Data Set Create utility (DBFUMRI0) 235
Control statements for the DBFUMRI0 utility . . . 237
Examples for the DBFUMRI0 utility 239

Chapter 19. HALDB Index/ILDS Rebuild utility (DFSPREC0) 241
Control statement for the DFSPREC0 utility . . . 244
Examples for the DFSPREC0 utility 245
Running the DFSPREC0 utility 245

Chapter 20. MSDB Dump Recovery utility (DBFDBDR0) 247
Control statements for the DBFDBDR0 utility. . . 251
Examples for the DBFDBDR0 utility 252

Part 4. Reorganization and conversion utilities 255

Chapter 21. Database Prereorganization utility (DFSURPR0) . 257
Control statements for the DFSURPR0 utility . . . 262
Examples for the DFSURPR0 utility 264

Chapter 22. HALDB Migration Aid utility (DFSMAID0) 267
Control statement for the DFSMAID0 utility . . . 271
Examples for the DFSMAID0 utility 272
Data spaces and sampling for the DFSMAID0 utility 273

Chapter 23. HD Reorganization Reload utility (DFSURGL0) 275
Utility control statements for the DFSURGL0 utility 283
Output Messages and Statistics for DFSURGL0 . . 284
Examples for DFSURGL0 286

Chapter 24. HD Reorganization Unload utility (DFSURGU0) 291
Control statements for the DFSURGU0 utility . . 301
Examples for the DFSURGU0 utility. 302

Chapter 25. High-Speed DEDB Direct Reorganization utility (DBFUHDR0) . . 309
Control statements for the DBFUHDR0 utility . . 313
Running the DBFUHDR0 utility 315

Chapter 26. HISAM Reorganization Reload utility (DFSURRL0). 319
Control statement for the DFSURRL0 utility . . . 325
Examples for the DFSURRL0 utility 326

Chapter 27. HISAM Reorganization Unload utility (DFSURUL0). 327
Control statements for the DFSURUL0 utility. . . 336
Examples for the DFSURUL0 utility 338

Chapter 28. MSDB-to-DEDB Conversion utility (DBFUCDB0) . . . 343
Control statements for the DBFUCDB0 utility . . 346
Examples for the DBFUCDB0 utility. 347
Converting an MSDB to a DEDB 350
Performing fallback from a DEDB to an MSDB . . 350
Exit routines for conversion and fallback for the DBFUCDB0 utility. 351

Chapter 29. Partial Database Reorganization utility (DFSPRCT1 and DFSPRCT2) 353
Control statements for the Partial Database Reorganization utility 359
Examples for the Partial Database Reorganization utility 362

Chapter 30. Utility Control Facility (DFSUCF00) 369
Control statements for the DFSUCF00 utility . . . 378
Examples for the DFSUCF00 utility 409
Running the DFSUCF00 utility 410

Part 5. Report and test utilities 419

Chapter 31. Database-Monitor Report Print utility (DFSUTR30) 421
Examples for the DFSUTR30 utility 423

Chapter 32. Program-Isolation-Trace Report utility (DFSPIRP0) 425
Control statement for the DFSPIRP0 utility . . . 428
Examples for the DFSPIRP0 utility 429

Chapter 33. SB Test utility (DFSSBHD0). 431
Control statements for the DFSSBHD0 utility. . . 436
Examples for the DFSSBHD0 utility 438

Part 6. Appendixes 439

Notices 441
Programming interface information 443
Privacy policy considerations 444

Bibliography 445

Index 447

About this information

These topics provide reference information for the utilities that you can use to migrate, reorganize, and recover a database.

This information is available in IBM® Knowledge Center at www.ibm.com/support/knowledgecenter.

Prerequisite knowledge

Before using the utilities described in this information, you should understand the fundamental concepts and tasks related to the z/OS® operating system. More importantly, you should understand IMS™ database concepts, including database organization, access methods, indexing methods, logical relationships, DEDB areas, HALDB partitioning, and so forth. You should also have a general understanding of such processes as database reorganization, conversion, back up, and recovery.

Most of the prerequisite IMS system and database information can be found in:

- *IMS Version 11 System Administration*
- *IMS Version 11 Database Administration*

You can learn more about z/OS by visiting the "z/OS basic skills education" topics in IBM Knowledge Center.

You can gain an understanding of basic IMS concepts by reading *An Introduction to IMS*, an IBM Press publication. An excerpt from this publication is available in the IBM Knowledge Center.

IBM offers a wide variety of classroom and self-study courses to help you learn IMS. For a complete list of courses available, go to the IMS home page at ibm.com/ims and link to the Training and Certification page.

IMS function names used in this information

In this information, the term HALDB Online Reorganization refers to the integrated HALDB Online Reorganization function that is part of IMS Version 11, unless otherwise indicated.

IMS 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 11, unless otherwise indicated.

How new and changed information is identified

New and changed information in most IMS library PDF publications is denoted by a character (revision marker) in the left margin. The first edition (-00) of *Release Planning*, as well as the *Program Directory* and *Licensed Program Specifications*, do not include revision markers.

Revision markers follow these general conventions:

© Copyright IBM Corp. 1974, 2015

- Only technical changes are marked; style and grammatical changes are not marked.
- If part of an element, such as a paragraph, syntax diagram, list item, task step, or figure is changed, the entire element is marked with revision markers, even though only part of the element might have changed.
- If a topic is changed by more than 50%, the entire topic is marked with revision markers (so it might seem to be a new topic, even though it is not).

Revision markers do not necessarily indicate all the changes made to the information because deleted text and graphics cannot be marked with revision markers.

How to read syntax diagrams

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

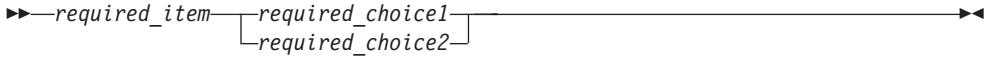
- Read the syntax diagrams from left to right, from top to bottom, following the path of the line. The following conventions are used:
  - The >>--- symbol indicates the beginning of a syntax diagram.
  - The ---> symbol indicates that the syntax diagram is continued on the next line.
  - The >--- symbol 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.

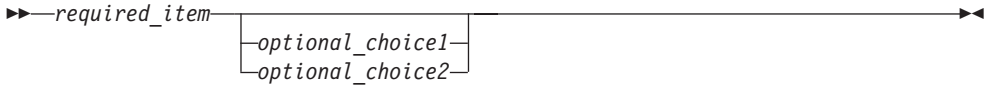
  If an optional item appears above the main path, that item has no effect on the execution of the syntax element and is used only for readability.

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

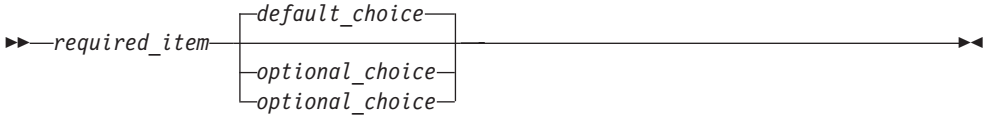
  If you *must* choose one of the items, one item of the stack appears on the main path.



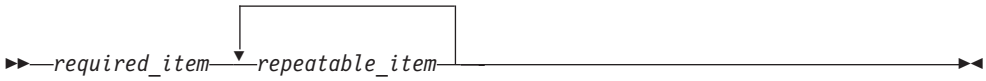
  If choosing one of the items is optional, the entire stack appears below the main path.



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



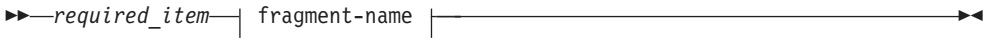
- An arrow returning to the left, above the main line, indicates an item that can be repeated.



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).

Accessibility features for IMS Version 11

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use information technology products successfully.

Accessibility features

The following list includes the major accessibility features in z/OS products, including IMS Version 11. These features support:

- Keyboard-only operation.
- Interfaces that are commonly used by screen readers and screen magnifiers.
- Customization of display attributes such as color, contrast, and font size.

Keyboard navigation

You can access IMS Version 11 ISPF panel functions by using a keyboard or keyboard shortcut keys.

For information about navigating the IMS Version 11 ISPF panels using TSO/E or ISPF, refer to the *z/OS TSO/E Primer*, the *z/OS TSO/E User's Guide*, and the *z/OS 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.

Related accessibility information

Online documentation for IMS Version 11 is available in IBM Knowledge Center.

IBM and accessibility

See the *IBM Human Ability and Accessibility Center* at www.ibm.com/able for more information about the commitment that IBM has to accessibility.

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:

- Click the **Feedback** link at the bottom of any IBM Knowledge Center topic.
- Send an email to imspubs@us.ibm.com. Be sure to include the book title and the publication number.

Changes to the IMS library

Changes to the IMS library for IMS Version 11 include minor title changes to all the publications, merging four sets of publications, a change to how *IMS Messages and Codes* are delivered, and a reorganization of *IMS Version 11 Exit Routines*. The library is no longer provided in BookManager format. In addition, as of the May, 2012, a master index is no longer provided.

The titles of all PDF publications in the IMS Version 11 library have been changed so that the publication titles match the contents view (navigation tree) titles in IBM Knowledge Center. For example, the *IMS Diagnosis Guide* has been renamed to *IMS Diagnosis*. In addition, the titles for the three *IMS Command Reference* PDF volumes have been renamed so that their titles better reflect the contents of each publication, as follows:

- *IMS Version 11 Commands, Volume 1: IMS Commands A-M*
- *IMS Version 11 Commands, Volume 2: IMS Commands N-V*
- *IMS Version 11 Commands, Volume 3: IMS Component and z/OS Commands*

Four sets of publications have been streamlined:

- To eliminate redundancy and group relevant application programming planning information with associated application programming tasks, the *IMS Application Programming Planning Guide* has been merged with the *IMS Application Programming Guide*, in a new title: *IMS Version 11 Application Programming*.
- To group diagnosis reference information with associated diagnosis tasks, the *IMS Diagnosis Reference* has been merged with the *IMS Diagnosis Guide*, in a new title: *IMS Version 11 Diagnosis*.
- To group all system administration tasks, regardless of whether the tasks relate to using a single IMS or an IMSplex, the *IMS IMSplex Administration Guide* has been merged with the *IMS System Administration Guide*, in a new title: *IMS Version 11 System Administration*.
- To group system definition reference information with associated system definition tasks, the *IMS System Definition Reference* has been merged with the *IMS System Definition Guide*, in a new title: *IMS Version 11 System Definition*.

IMS Version 11 Exit Routines is now organized by the call interface and available services for each group of exit routines, as follows:

- IMS system exits, database manager exits, and transaction manager exit routines are grouped under “IMS control region exit routines.”These exit routines use IMS interfaces and callable services, as described in the topic “Overview of exit routines for IMS.”
- Exit routines that run in an address space with BPE support, including those for CQS, BPE-based DBRC, and most CSL components, are grouped under “Base Primitive Environment-based exit routines.” BPE-based exit routines use the interfaces and services provided by the Base Primitive Environment, as described in the topic “BPE user-supplied exit routine interfaces and services.”
- The CQS exit routines are now divided into two groups: those that run in the CQS address space and have access to BPE services are grouped under “BPE-based CQS user-supplied exit routines,” and those that execute in a CQS client address space are grouped under “CQS client exit routines.”

© Copyright IBM Corp. 1974, 2015

- Standard DBRC exit routines are grouped under “IMS system exit routines,” while the variants that run in BPE are group under “BPE-based DBRC exit routines.”
- CSL SCI exit routines are now divided into two categories: those that run in the CSL SCI address space are grouped under “BPE-based SCL SCI user exit routines,” and those that run in the address space of a registered IMSplex member are grouped under “CSL SCI IMSPlex member exit routines.”
- Exit routines that have specific guidelines, including those for IMS Connect, CQS clients, and CSL SCI IMSPlex members, are described in the sections for those routines.

Part 1. Definition and initialization utilities

Use the definition and initialization utilities to define Fast Path databases and full-function databases, including Fast Path DEDB databases and full-function HALDB partitioned databases.

© Copyright IBM Corp. 1974, 2015

Chapter 1. Database Prefix Resolution utility (DFSURG10)

Use the Database Prefix Resolution utility (DFSURG10) to accumulate the information that is generated on work data sets during the load, reorganization, or both of one or more databases.

The DFSURG10 utility produces an output data set that contains the prefix information that is needed to complete the logical relationships that are defined for the databases. The utility can also produce an output data set containing information that is needed to create or update secondary index databases.

The DFSURG10 utility requires a control data set that is created by running the Database Prereorganization utility (DFSURPR0) and does not use utility control statements.

Prefix resolution is not required for HALDB databases.

The functions of this utility also can be performed by the Utility Control Facility (DFSUCF00).

The following is a flow diagram of the Database Prefix Resolution utility.

© Copyright IBM Corp. 1974, 2015

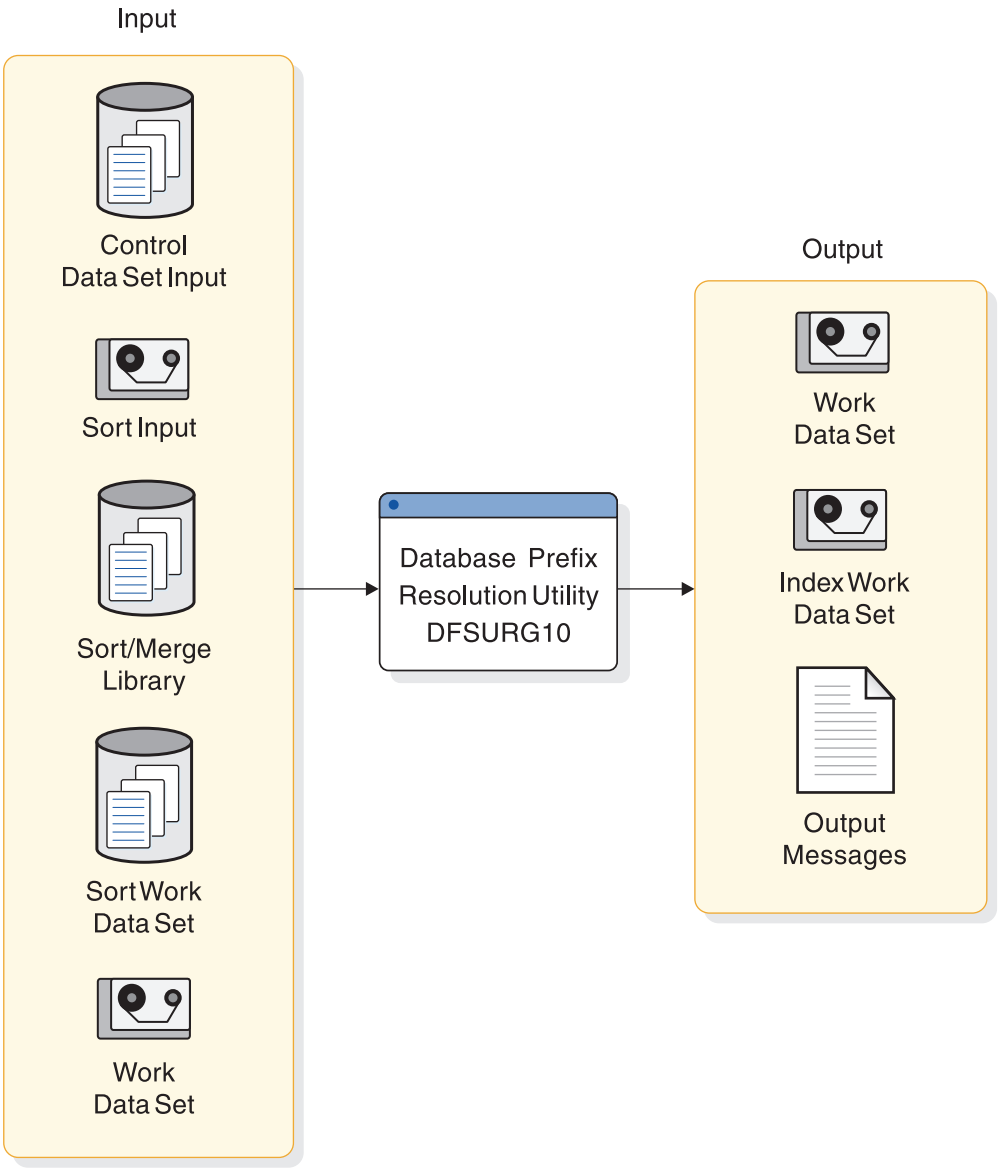


Figure 1. Database Prefix Resolution utility

Subsections:

- "Restrictions"
- "Prerequisites" on page 5
- "Requirements" on page 5
- "Recommendations" on page 6
- "Input and output" on page 6
- "Return codes" on page 6
- "JCL specifications" on page 7

Restrictions

The DFSURG10 utility uses the z/OS SORT/MERGE program. Because the maximum sort field permitted by the z/OS SORT/MERGE program 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 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 later by an application program operating in update mode. After 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.

IMS makes no assumption of sequence for unkeyed or non-unique keyed segments during initial database load, and the z/OS SORT/MERGE program 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.

The DFSURG10 utility works on full-function non-HALDB databases only.

Utilities that alter databases cannot be run while the database is quiesced.

Prerequisites

Before you run the DFSURG10 utility, you must first run other application programs or utilities to generate the data sets that the DFSURG10 utility uses as input. The application programs and utility programs that you must run and the data sets they create include:

- The Database Prereorganization utility (DFSURPR0), to generate a control data set used by the DFSURG10 utility.
- A load, reorganization, or scan application program or utility, to create the required DFSURWF1 work data set that is used by the DFSURG10 utility. The type of application program or utility that you run to create the DFSURWF1 work data set depends on the task you must complete.

Requirements

Currently, no requirements are documented for the DFSURG10 utility.

Recommendations

When specifying a size on the EXEC statement of the DFSURG10 utility, estimate a file size if possible because this improves SM1 optimization and performance.

Input and output

The primary inputs to the DFSURG10 utility are a work data set that is created by a load, reorganization, or scan application program or utility and a control data set that is created by the Database Prereorganization utility (DFSURPR0). The DFSURG10 utility also uses other optional and required input.

The DFSURWF1 work data sets submitted to the DFSURG10 utility must be concatenated to form a 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.

The primary output of the DFSURG10 utility is a work data set used by the Database Prefix Update utility (DFSURGP0) and, optionally, a work data set used by the HISAM Unload program (DFSURUL0) for creating, replacing, merging, or extracting secondary indexes. The DFSURG10 utility also produces messages and statistics.

The following table identifies inputs and outputs that are used by the Database Prefix Resolution utility.

Table 1. Data sets used by the Database Prefix Resolution utility

Input	Output
Control data set	Work data set
Sort input	Messages
Sort merge library	Statistics, if STAT or SUMM is specified in the Database Prereorganization utility (DFSURPR0) control statement
Sort work data set	
Work data set	
Index work data set	

Return codes

The following return codes are provided at program termination:

Code	Meaning
0	No errors were detected.
4	Returned when any of the following messages have been issued during program execution or the following messages have been preempted by the SUMM parameter during DB Prereorganization: DFS878, DFS885, DFS961.
8	Returned when data required by prefix update is not written to the WF3 data set, or when one or more of the following messages has been issued during program execution: DFS852, DFS855, DFS857, DFS876, DFS877, DFS879, DFS880, DFS881.

12 Returned when either one or both of the messages listed under return code 4 and any one or more of the messages listed under return code 8 have been issued.

16 or higher
Returned by the 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 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.

JCL specifications

The DFSURG10 utility is executed as a standard z/OS job. The JCL specifications for the DFSURG10 utility include a JOB statement, the EXEC statement, and the DD statements. The DFSURG10 utility does not use utility control statements.

The following JCL statements are required:

- A JOB statement that you define to meet the specifications of your installation.
- An EXEC statement
- DD statements that define inputs and outputs

You can use the IMS-supported PARM field options of EXEC statement to specify options for the z/OS SORT/MERGE program used by the DFSURG10 utility.

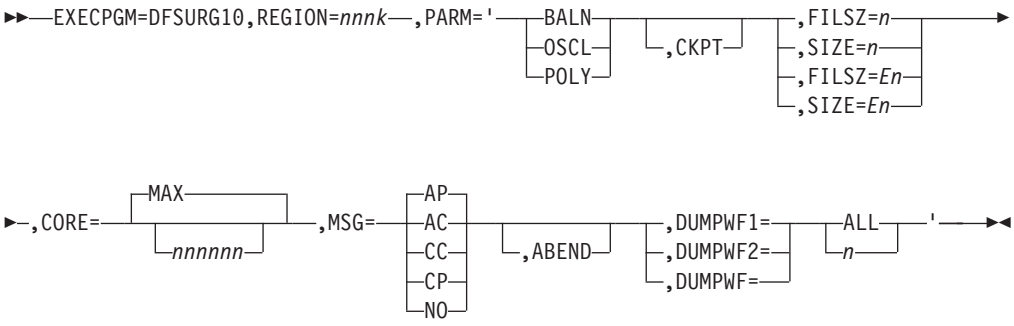
EXEC statement

The EXEC statement must be in the form that is shown in the following example:

```
//STEP EXEC PGM=DFSURG10,REGION=100K,PARM='CKPT,MSG=AC,FILSZ=E5000'
```

The Database Prefix Resolution utility invokes a separate sort program. Increasing the size specified on the REGION= parameter can improve the performance of some sort programs.

The PARM field specifications are not positional. The following syntax diagram shows all of the PARM field options and their default specifications:



Only the keywords shown in the syntax diagram are passed to the attached z/OS SORT/MERGE program.

ABEND
Specify only when a storage dump is required for diagnostic purposes. When specified, a SYSUDUMP DD statement is also required.

CKPT
Specifies that the DFSURG10 utility links to the z/OS SORT/MERGE program instead of attaching.

CORE
Specifies the amount of main storage used by the z/OS SORT/MERGE program as a six-digit figure. Calculates the value for CORE= based on the type of sort and the sort devices used.

The default value for CORE= is MAX, which does not place a limit on the amount of main storage that the z/OS SORT/MERGE program uses.

Recommendation: If you do not want to use the default specification of CORE=MAX and you are unsure about what to specify, use the minimum limit that is required for Data Facility Sort (DFSORT). If you are using an OEM sort product that allocates unit control blocks (UCBs) below the 16 MB line, use CORE=MAX.

DUMPWF1, DUMPWF2, and DUMPWF

Diagnostic features 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 nine digits.

FILSZ
If FILSZ=*n* is specified, *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.

If FILSZ=E*n* is specified, *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 z/OS 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 z/OS 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, estimate a file size, because this improves the optimization and performance of SM1.

MSG
Controls the messages printed by the z/OS SORT/MERGE program. The default specification of AP prints all DFSORT messages and sends only critical messages to the z/OS master console.

OSCL
Specifies the Oscillating technique for tape work data sets. If OSCL is specified, a SIZE parameter must be specified.

DFSORT accepts this EXEC PARM options, but does not process it.

SIZE
If SIZE=*n* is specified, *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.

If SIZE=E*n* is specified, *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 z/OS 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 z/OS 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.

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.

The 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 z/OS SORT/MERGE program. 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 z/OS SORT/MERGE program.

This DD statement is required.

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

This DD statement is required.

SORTIN DD
Defines the input data set for this program. It is referenced by the z/OS 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.

The 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.

This DD statement is required.

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.

The 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.

The 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 Database Prereorganization utility (DFSURPR0).

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).

The DCB parameters specified within this program are RECFM=VB and LRECL=900. BLKSIZE must be provided on the DFSURIDX DD statement.

This DD statement is required only if secondary indexes are present.

Related reference:
Chapter 30, "Utility Control Facility (DFSUCF00)," on page 369

Examples for the DFSURG10 utility

This example shows usage for the DFSURG10 utility.

The example below uses the Database Prefix Resolution utility to resolve logical relationships and secondary indexes. Only three work data sets are supplied to the z/OS SORT/MERGE program and it 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. 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.

Figure 2. Resolving logical relationships and secondary indexes with the Database Prefix Resolution utility

```
//PREFXRES EXEC PGM=DFSURG10,REGION=100K,PARM='CKPT,MSG=AC,FILSZ=E5000'
//SYSUDUMP DD SYSOUT=A
//SYSPRINT DD SYSOUT=A,DCB=BLKSIZE=1200
//SYSOUT   DD SYSOUT=A
//SORTLIB  DD DSN=SYS1.SORTLIB,DISP=SHR,VOL=SER=SYSLIB,UNIT=SYSDA
//SORTWK01 DD UNIT=SYSDA,SPACE=(1008,(60),,CONTIG)
//SORTWK02 DD UNIT=SYSDA,SPACE=(1008,(60),,CONTIG)
//SORTWK03 DD UNIT=SYSDA,SPACE=(1008,(60),,CONTIG)
//SORTIN   DD DSN=&&WF1,UNIT=SYSDA,DISP=(MOD,PASS)
//         DD DSN=&&WWF1,UNIT=SYSDA,DISP=(MOD,PASS)
//DFSURWF2 DD DSN=&&WF2,UNIT=SYSDA,SPACE=(1008,(30),,CONTIG),
//         DISP=(,PASS),DCB=(RECFM=VB,LRECL=900,BLKSIZE=1008)
//DFSURWF3 DD DSN=&&WF3,UNIT=SYSDA,SPACE=(1008,(30),,CONTIG),
```

```
//         DISP=(,PASS),DCB=(RECFM=VB,LRECL=900,BLKSIZE=1008)
//DFSURCDS DD DSN=*.LDJJK310.PREREORG.DFSURCDS,
//            UNIT=SYSDA,DISP=(OLD,PASS),
//            VOL=REF=*.LDJJK310.PREREORG.DFSURCDS
//DBHVSAM1 DD DSN=DIVNTZ04.JJXXS01K,DISP=SHR
//DBHVSAM2 DD DSN=DIVNTZ04.JJXXS01E,DISP=SHR
//HIDAM    DD DSN=DHONTZ04.JKXXI01O,DISP=SHR
//HIDAM2   DD DSN=DHONTZ04.JKXXI02O,DISP=SHR
//XDLBT04I DD DSN=DXINTZ04.JKXXS01I,DISP=SHR
/*
```

Chapter 2. Database Prefix Update utility (DFSURGP0)

Use the Database Prefix Update utility (DFSURGP0) to update the prefix of each segment whose prefix information was affected by a database load, reorganization, or both.

The DFSURGP0 utility uses the output that is generated by the Database Prefix Resolution utility (DFSURG10) to update the segment prefixes.

The DFSURGP0 utility updates the following prefix fields:

- Logical parent pointer fields
- Logical twin pointer fields
- Logical child pointer fields
- Counter fields associated with logical parents

HALDB databases do not require you to update segment prefixes because they use a self-healing pointer process that updates the segment prefixes.

The DFSURGP0 utility can be restarted after abnormal terminations; however, the restart process differs depending on whether the abnormal termination was due to a database I/O error.

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

The following figure is a flow diagram of the Database Prefix Update utility.

© Copyright IBM Corp. 1974, 2015

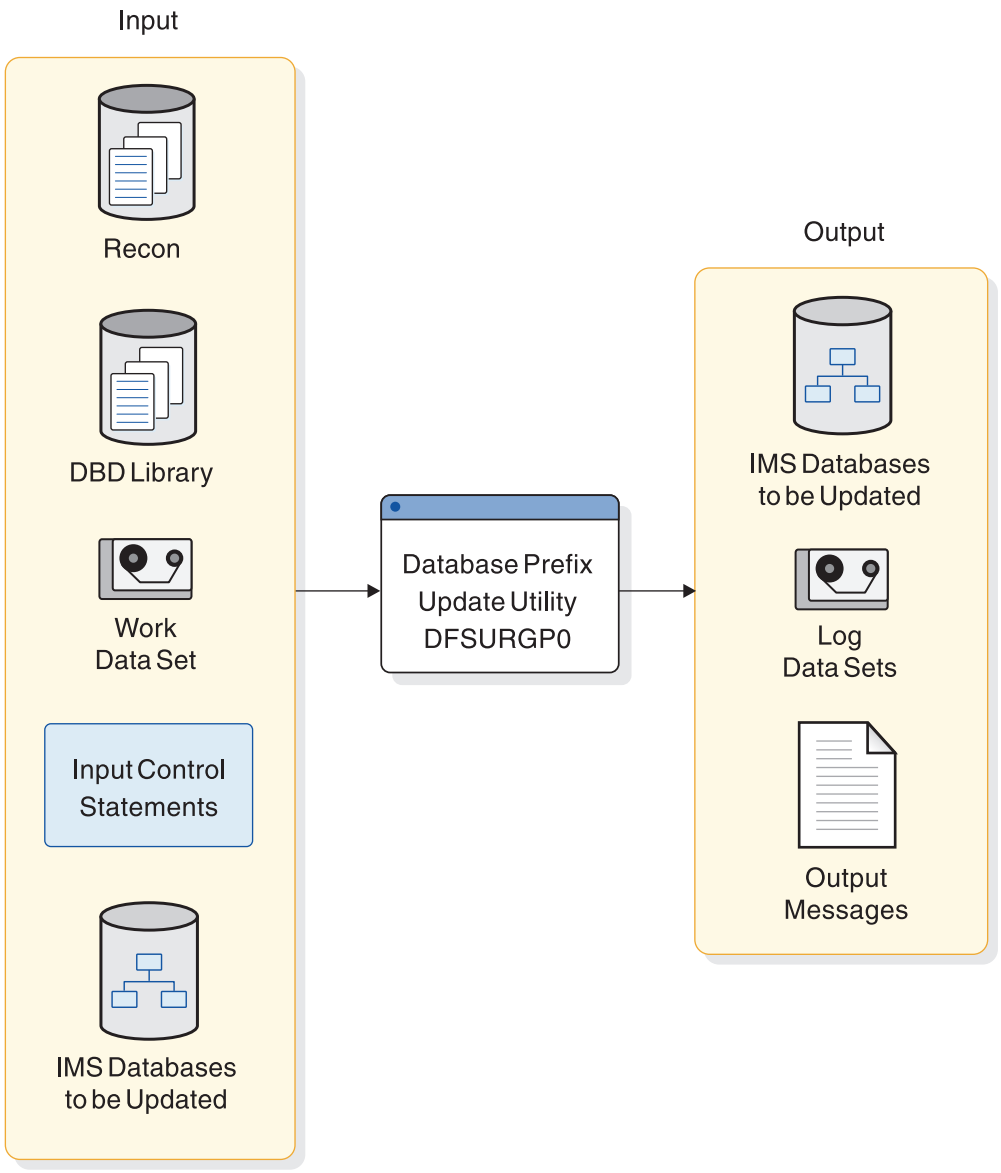


Figure 3. Database Prefix Update utility

Subsections:

- "Restrictions"
- "Prerequisites" on page 15
- "Requirements" on page 15
- "Recommendations" on page 15
- "Input and output" on page 15
- "Return codes" on page 15
- "JCL specifications" on page 16

Restrictions

The DFSURGP0 utility will work on full-function non-HALDB databases only.

Utilities that alter databases cannot be run while the database is quiesced.

Prerequisites

Before you run the DFSURGP0 utility, you must run the Database Prefix Resolution utility (DFSURG10) to generate the input for the DFSURGP0 utility.

Requirements

Currently, no requirements are documented for the DFSURGP0 utility.

Recommendations

Currently, no recommendations are documented for the DFSURGP0 utility.

Input and output

The primary input to the DFSURGP0 utility is a work file that is created by the Database Prefix Resolution utility (DFSURG10). The primary output of the DFSURGP0 utility is a database data set in which all of the segment prefixes that are used for logical relationships are correct.

The following table identifies inputs and outputs used by the Database Prefix Update utility.

Table 2. 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	

The input to the DFSURGP0 utility is one or more update records to be applied to each segment that contains logical relationship prefix information. The update records are sorted into database and segment physical location order by the Database Prefix Resolution utility.

Optionally, the DFSURGP0 utility can create log output data sets if you include log DD statements (IEFRDER, IEFRDER2) in the JCL. You can use this log output for database recovery. The log output is especially useful for the update of scanned databases. However, batch backout cannot be performed by 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 the RECON data set for each data set updated.

If the DFSURGP0 utility terminates without errors, the utility issues a normal program termination message that indicates the number of records processed.

Return codes

The following return codes are provided at program termination:

Code	Meaning
0	No errors were detected.

8 One or more error messages were issued. The messages contain details on the errors and are printed as part of the system output.

JCL specifications

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

- A JOB statement that you define to meet the specifications of your installation.
- 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.

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.

The STEPLIB DD statement is required.

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

The DFSRESLB DD statement is required only if an unauthorized library is concatenated to IMS.SDFSRESL.

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.

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.

The SYSIN DD statement is required only if you include utility control statements as input to the DFSURGP0 utility.

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.

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).

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.

Related concepts:

IMS buffer pools (System Definition)

Related reference:

Utility Control Facility (DFSUCF00) (Database Utilities)

Sequential buffering control statements (System Definition)

DBBBATCH procedure (System Definition)

DLIBATCH procedure (System Definition)

Control statements for the DFSURGP0 utility

You can use four utility control statements with the DFSURGP0 utility: CHKPT, RSTRT, SNAP, and ABEND.

CHKPT statement

The CHKPT utility control statement indicates that utility checkpoint operations are to be performed during operation of this program. The Database Prefix Resolution utility (DFSURG10) automatically writes records on the DFSURWF3 data set as a service for the DFSURGP0 utility, and these records are used as utility checkpoints. As each utility 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 utility checkpoint record, and the volume serial identifier of the volume being processed. Save the utility checkpoint messages in case a restart operation is required.

RSTRT statement

The RSTRT 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 that contains the checkpoint record numbered *nnnnn.* The *nnnnn* and *volser* parameters 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

The SNAP 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

The ABEND utility control statement indicates that 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.

Examples for the DFSURGP0 utility

These examples show sample JCL for the DFSURGP0 utility.

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
```

Subsections:

- "Update databases" on page 20
- "Two logically related databases reorganization" on page 20

Update databases

The following figure shows the JCL required to execute DFSURGP0 to update the five databases defined by the DBHVSAM1, DBHVSAM2, HIDAM, HIDAM2, and XDLBT04I DD statements.

Figure 4. Example of updating five databases

```
//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)
/*
```

Two logically related databases reorganization

The following figure 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).

Figure 5. Example of reorganizing two logically related databases

```
//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'
//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)
 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
//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)
```

```
//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
/*
//
```

Restarting after an abnormal termination of the DFSURGP0 utility

If the DFSURGP0 utility abnormally terminates 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 the DFSURGP0 utility abnormally terminates because of a database I/O error, perform the following actions:

1. Determine the cause of the I/O error and rectify it.
2. Restore the database as it existed before execution of this program.
3. Either request a restart operation or re-run the program using the original input work data set.

Chapter 3. Database Scan utility (DFSURGS0)

Use the Database Scan utility (DFSURGS0) to scan non-HALDB databases that are not being loaded or reorganized.

During the utility identifies segments that contain 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 DFSURGS0 utility writes records to the DFSURWF1 output work data set. The DFSURWF1 work data set is used as one of the inputs to the Database Prefix Resolution utility (DFSURG10).

The DFSURGS0 utility can be restarted after abnormal terminations; however the restart process differs depending on whether or not the abnormal termination was due to a database I/O error.

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

The following figure shows a flow diagram of the Database Scan utility.

© Copyright IBM Corp. 1974, 2015

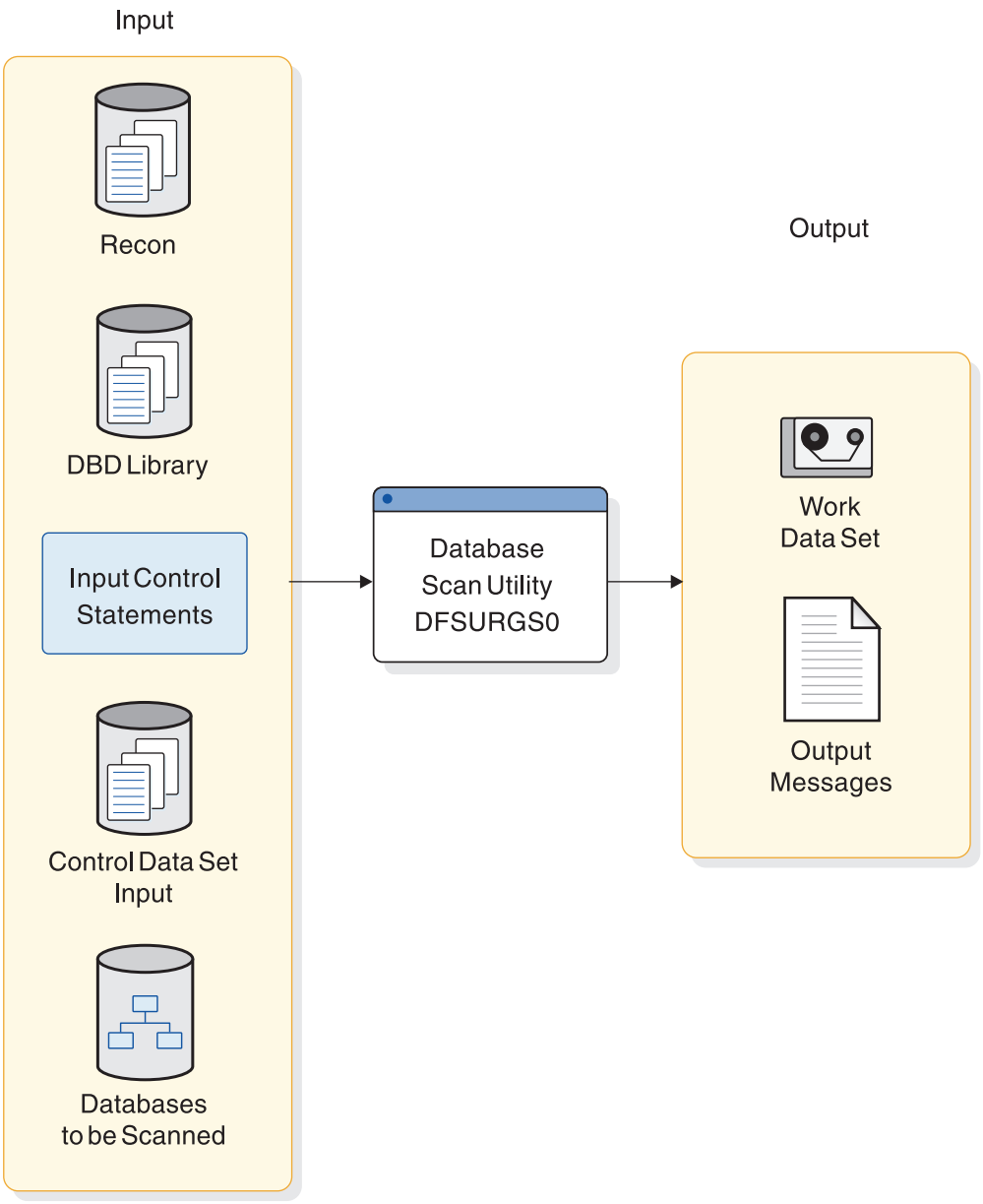


Figure 6. Database Scan utility flow diagram

Subsections:

- "Restrictions" on page 27
- "Prerequisites" on page 27
- "Requirements" on page 27
- "Recommendations" on page 27
- "Input and output" on page 27
- "Return codes" on page 27
- "JCL specifications" on page 27

Restrictions

The DFSURGS0 utility will work on full-function non-HALDB databases only.

Utilities that alter databases cannot be run while the database is quiesced.

Prerequisites

Currently, no prerequisites are documented for the DFSURGS0 utility.

Requirements

Currently, no requirements are documented for the DFSURGS0 utility.

Recommendations

Currently, no recommendations are documented for the DFSURGS0 utility.

Input and output

The primary input to the DFSURGS0 utility is one or more databases to be scanned. The DFSURGS0 utility accepts other optional and required input.

The primary output of the DFSURGS0 utility is a work data set. The DFSURGS0 utility also produces output messages.

The following table identifies inputs and outputs for the Database Scan utility.

Table 3. Inputs and outputs for the Database Scan utility

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

Return codes

The following return codes are provided at program termination:

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

JCL specifications

The DFSURGS0 utility is executed as a standard z/OS job. The JCL specifications for the DFSURGS0 utility include a JOB statement, the EXEC statement, and the DD statements. One or more utility control statements must be/can be included with the JCL statements.

The following JCL statements are required:

- A JOB statement that you define to meet the specifications of your installation.

- 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.

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.

The DFSRESLB DD statement is required only if an unauthorized library is concatenated to IMS.SDFSRESL.

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.

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).

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.

Related concepts:

IMS buffer pools (System Definition)

Related reference:

Chapter 30, "Utility Control Facility (DFSUCF00)," on page 369

Sequential buffering control statements (System Definition)

DBBBATCH procedure (System Definition)

DLIBATCH procedure (System Definition)

Control statements for the DFSURGS0 utility

You can use four utility control statements for the DFSURGS0 utility: DBS, CHKPT, RSTRT, and ABEND.

DBS statement

The DBS 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 eight 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

The CHKPT utility control statement indicates that utility checkpoint operations are to be performed during operation of this program.

A utility 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 utility checkpoint record is generated, a utility checkpoint message (DFS867I) is issued to the z/OS system console. The message indicates the name of this program, the checkpoint number of the utility checkpoint record written, and the output volume serial of the volume being written. Save the utility checkpoint messages in case a restart operation is required.

RSTRT statement

The RSTRT utility control statement indicates that a restart operation is to be performed by this program.

The *nnnnn* is a five-digit decimal number and *volser* is the volume serial identifier of the input volume that contains the checkpoint record numbered *nnnnn*. The *nnnnn* and *volser* parameters are 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

Use the ABEND utility control statement when a storage dump is required for diagnostic purposes.

If this utility control statement is 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.

Examples for the DFSURGS0 utility

These examples show sample JCL for the DFSURGS0 utility.

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

Figure 7. DD statements for using DBRC without dynamic allocation

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

The following figure 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 rather than control statements.

Figure 8. Scanning a database defined by the HDRELTD DD statement

```
//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
```

DD statements must be included for all logically related databases.

Restarting after an abnormal termination of the DFSURGS0 utility

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, perform the following actions:

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.

Chapter 4. Database Surveyor utility (DFSPRSUR)

Use the Database Surveyor utility (DFSPRSUR) to scan all or part of an HDAM or HIDAM database and to produce a report describing the physical organization of the database.

This report can help you determine the need for reorganization. The Database Surveyor utility also identifies the size and location of free space areas that can receive reorganized records during a partial database reorganization.

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

Subsections:

- "Restrictions"
- "Prerequisites"
- "Requirements"
- "Recommendations"
- "Input and output"
- "Return codes" on page 36
- "JCL specifications" on page 36

Restrictions

The DFSPRSUR utility will work on full-function non-HALDB databases only.

Utilities that alter databases cannot be run while the database is quiesced.

Prerequisites

Currently, no prerequisites are documented for the DFSPRSUR utility.

Requirements

If the DFSPRSUR utility executes as a batch program in a utility batch region, the DFSVSAMP DD statement is required. When the DFSPRSUR utility runs as a BMP, the buffers are provided by the VSAM buffer pool manager in the IMS DLISAS control region.

Recommendations

Currently, no recommendations are documented for the DFSPRSUR utility.

Input and output

The database to be analyzed can be shared when the DFSPRSUR utility is executing as a BMP. Input utility control statements direct the DFSPRSUR utility to specific sections (key ranges or block number ranges) of a particular database.

A PSB containing a PCB for the database being surveyed must be defined for the use by the DFSPRSUR utility. (More than one database PCB can be contained in

© Copyright IBM Corp. 1974, 2015

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 DFSPRSUR utility 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 DFSPRSUR utility produces a Database Surveyor report as output. For each partition of a range specified, the Database 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.

Return codes

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

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

JCL specifications

The DFSPRSUR utility is executed as a standard z/OS job. The JCL specifications for the DFSPRSUR utility include a JOB statement, the EXEC statement, and the DD statements. One or more utility control statements must be/can be included with the JCL statements.

The following JCL statements are required:

- A JOB statement that you define to meet the specifications of your installation.
- 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.

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.

The DFSVSAMP DD statement is required when the DFSPRSUR utility executes as a batch program in a utility batch region. When the DFSPRSUR

utility runs as a BMP, the buffers are provided by the VSAM buffer pool manager in the IMS DLISAS control region.

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

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.

Related concepts:

IMS buffer pools (System Definition)

Related reference:

Chapter 29, "Partial Database Reorganization utility (DFSPRCT1 and DFSPRCT2)," on page 353

Sequential buffering control statements (System Definition)

DBBBATCH procedure (System Definition)

DLIBATCH procedure (System Definition)

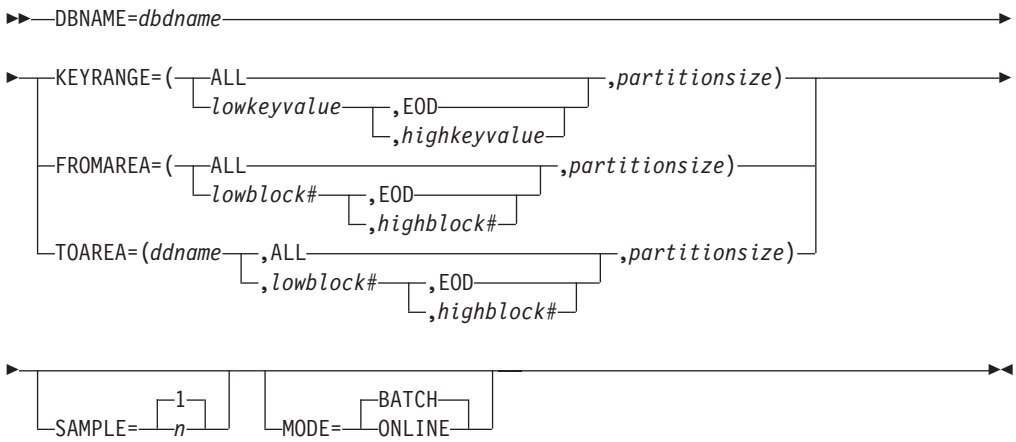
Control statement for the DFSPRSUR utility

The DFSPRSUR utility uses input statements to describe processing options.

An input statement must conform to the following guidelines:

- 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 10 TOAREA definitions can be specified for a single database.

Note: 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 on, 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.

Examples for the DFSPRSUR utility

The examples in this topic show sample JCL for the DFSPRSUR utility.

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 the figure below to the sample JCL:

Figure 9. DD statements for using DBRC without dynamic allocation

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

Subsections:

- "Execute DFSPRSUR as a batch" on page 41
- "Execute DFSPRSUR as a batch program JCL" on page 43

• "Execute DFSPRSUR as a BMP" on page 45

Execute DFSPRSUR as a batch

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
```

The figure below is a sample FROMAREA partition report produced by running Example 1. One of these reports is produced for each partition.

Figure 10. Surveyor-FROMAREA-Partition report

```
     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
```

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

Figure 11. Surveyor-FROMAREA-Range report

```
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
```

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.

Execute DFSPRSUR as a batch program JCL

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
```

The figure below is a sample partition report produced running Example 2. One of these reports is produced for each partition.

Figure 12. Surveyor-TOAREA-Partition report

```
     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
```

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.

The figure below is a sample range report produced by running Example 2. One of these reports is produced to summarize the entire range.

Figure 13. Surveyor-TOAREA-Range report

```
     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
```

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.

Execute DFSPRSUR as a BMP

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
```

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

Figure 14. Surveyor-KEYRANGE-Partition report

```
  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
```

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.

The figure below is a sample range report produced by running Example 3. One of these reports is produced to summarize the entire range.

Figure 15. Surveyor-KEYRANGE-Range report

```
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
```

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 5. DEDB Initialization utility (DBFUMIN0)

Use the DEDB Initialization utility (DBFUMIN0) to initialize one or more active data sets of one or more areas of a DEDB database.

- 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.
- After the data has been loaded, create an image copy of the database. If a system failure occurs before the data is accessed, the image copy is needed for recovery. If an image copy is not taken before accessing the data and a failure occurs, the data set must be redefined and the DEDB Initialization utility must be rerun.

Subsections:

- "Restrictions"
- "Prerequisites"
- "Requirements"
- "Recommendations" on page 50
- "Input and output" on page 50
- "Return codes" on page 50
- "JCL specifications" on page 51

Restrictions

The following restrictions apply when using DBFUMIN0:

- This utility cannot be stopped and restarted. It must be rerun from the beginning.
- The Online Database Image Copy utility does not support DEDBs, and there is no facility for recovering the DEDB databases until a batch image copy is created.
- 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.
- Because only one DEDB database can be specified as a member in an ACBLIB DD statement, only the areas of one DEDB database can be initialized at a time.
- The DBFUMIN0 utility will work on Fast Path (DEDB) databases only.
- Primary space allocation may be split across multiple volumes, but secondary allocation and dataset extension are not supported.

Prerequisites

The DEDB database must be allocated using the z/OS IDCAMS command.

Requirements

To run the DBFUMIN0 utility you must satisfy various operational requirements.

The following operational requirements apply when using the DEDB Initialization utility:

© Copyright IBM Corp. 1974, 2015

- To initialize the multiple area data sets of an area, the status of the area data sets must be unavailable in the RECON data set.
- When there are multiple area data sets of an area, all area data sets must be initialized with DBFUMIN0 before opening. To add additional area data sets, use the Data Set Create utility (DBFUMRI0) before opening the area after IMS has been started.
- 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. The DBFUMIN0 utility does not format any portion of the area that resides on a candidate volume.

Requirement: A DD statement is required for each area and for each ADS to be initialized.

Recommendations

Currently, no recommendations are documented for the DBFUMIN0 utility.

Input and output

The DEDB Initialization utility uses the following input:

- The ACB generated for the specific DEDB (access to the DBD member of ACBLIB is required)
- DBRC RECON data set if the area is registered and is in recovery-needed status
- DBRC RECON data set if the area is registered and the area data set is in unavailable status
- A control data set that specifies which areas are to be initialized

The utility produces the following output:

- A data set that contains output messages and statistics
- 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

Return codes

The DEDB Initialization utility provides one of the following return codes:

Code	Meaning
0	Initialization was successful.
4	Error in parameters.
8	ACB or DBRC processing error.
12	Error in processing data set information.
16	Error during format processing.
20	SYSPRINT error.

JCL specifications

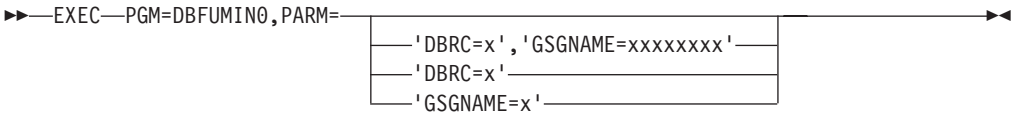
The DBFUMIN0 utility is executed as a standard z/OS job. The JCL specifications for the DBFUMIN0 utility include a JOB statement, the EXEC statement, and the DD statements. One or more utility control statements must be included with the JCL statements.

The following JCL statements are required:

- An EXEC statement
- DD statements that define the inputs and outputs

EXEC statement

The EXEC statement must be in the form:



The DBNAME from the DMCB is used as the password for the DEDB areas.

When the DBRC=N parameter is specified in the EXEC statement, the RECON*n* DD statements are not required.

When the DBRC=Y parameter is specified in the EXEC statement, the RECON*n* 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 sample installation default member DFSIDEF0. The default for the batch subsystem in the DFSIDEF0 member is DBRC=YES. If you do not use the DFSIDEF0 sample module or if the module was not loaded at initialization time, IMS defaults to DBRC=YES. If the installation defaults module DFSIDEF0 has DBRC=FORCE, then the error message DFS0044I DBRC REQUIRED FOR THIS EXECUTION displays.

The GSGNAME parameter specifies the RSR global service group name if RSR is used.

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 about 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.

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.

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.

Related concepts:

Initializing and maintaining the RECON data sets (System Administration)

Control statement for the DBFUMIN0 utility

The AREA 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.

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.

If you are specifying areas, you can specify only one area on a control statement. However, you can specify more than one area by using multiple control statements.

AREA

areaname
The name of the area to be initialized. Enter the area name exactly as it is defined on the DD1 parameter of the AREA statement in the DBD source.

ALL
Specifies that all areas in the DEDB are to be initialized. If ALL is specified, then the following restrictions apply:

- ALL must be specified at the first record in the CONTROL data set.

- The remainder of this control statement and all following control statements are ignored.

Examples of the DBFUMIN0 utility

These examples show sample JCL for the DBFUMIN0 utility.

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 the figure below to the sample JCL:

Figure 16. DD statements for using DBRC without dynamic allocation

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

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 6. DEDB Sequential Dependent Delete utility (DBFUMDL0)

Use the DEDB Sequential Dependent Delete utility (DBFUMDL0) online to logically delete sequential dependent segments within a specified limit of a DEDB area.

After the dependent segments are deleted, the utility resets the segment boundaries, and the freed space in the area is available for reuse.

The DBFUMDL0 utility runs in the FP utility type dependent region, as if it were an application executing in an IFP region.

Subsections:

- "Restrictions"
- "Prerequisites"
- "Requirements"
- "Recommendations"
- "Input and output" on page 56
- "Return codes" on page 56
- "JCL specifications" on page 56

Restrictions

You must execute the DEDB Sequential Dependent Delete utility in a separate job step. The program cannot switch from one utility or one database to another within the same job step.

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 that is defined by the restart parameter.

The DBFUMDL0 utility will work on Fast Path (DEDB) databases only.

Utilities that alter databases cannot be run while the database is quiesced. .

Prerequisites

Currently, no prerequisites are documented for the DBFUMDL0 utility.

Requirements

Currently, no requirements are documented for the DBFUMDL0 utility.

Recommendations

Currently, no recommendations are documented for the DBFUMDL0 utility.

© Copyright IBM Corp. 1974, 2015

Input and output

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

The DEDB Sequential Dependent Delete utility produces the following output:

- An area with freed space
- A data set that contains output messages and statistics

Return codes

Currently, no return codes are documented for the DBFUMDL0 utility.

JCL specifications

The DBFUMDL0 utility is executed as a standard z/OS job. One or more utility control statements must be included with the JCL statements.

The following JCL statements are required:

- A JOB statement
- An EXEC statement
- DD statements that specify the 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
```

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 DEDB utility commands that control the processing options of the DBFUMDL0 utility.

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

Related reference:

FPUTIL procedure (System Definition)

Control statements for the DBFUMDL0 utility

The DBFUMDL0 utility uses DEDB utility commands to define its processing options. The DEDB utility commands are specified in a control data set that is allocated to the DBFUMDL0 utility by the SYSIN DD statement.

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.

The SYSIN data set can contain the following DEDB utility commands:

ALLFMNEW (optional)
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. 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 (required)
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 - 8 characters. The first character must be alphabetic, and the rest of the characters are alphanumeric. Alphabetic characters include @, #, and $.

BUFNO (optional)
Specifies the number of buffers to use to read or write the DEDB. For the utilities other than the Reorganization utility, a minimum of seven 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.

ERRORACTION (optional)
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.

GO (optional)
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.

LASTSCAN (optional)
Acts as a substitute for a stop value, provided that no other utility was run on the area except a prior SCAN. LASTSCAN uses the stopping point of the SCAN as the stopping point of DELETE.

NSQCI (optional)
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 (optional)
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 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 (optional)
Specifies a stop time in hex format. You can use HEX time stamps to specify much more detail. The HEX time stamp of the newest SDEP SEGMENT can be obtained using the POS call.

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

```
STOPHEXT  X'C5C55A5DB883FE33'   /* 16 hex digits required.  */
Note:  C5C55A5DB883FE33   =   2010.092 18:44:05.131.327.875
```

STARTHEXT and STOPHEXT use the same time stamp format and are much more detailed than STARTIME and STOPTIME.

STOPRBA **(optional)**
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 Database Scan utility extracts SDEPS up to, but excluding the SDEP segment at STOPRBA , while the Database Delete utility sets field DMACLBTS to STOPRBA rather than to STOPRBA+1.

STOPROOT **(optional)**
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 (optional)
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 - 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.

STOPTIME (optional)
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=required offset to UTC which when added to the UTC gives local time. 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>
STOPTIME=C'2010.092 18:40:05 -00:00'
```

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

TYPE (required)
Specifies either the scan, delete, or reorganization utility as the type of run.

Attention: 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.

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.

Related reference:

"Control statements for the DBFUMSC0 utility" on page 66

Examples for the DBFUMDL0 utility

These examples show sample JCL for the DBFUMDL0 utility.

Sample JCL for the Database Scan utility

The following figure shows the:

- JCL for the Database Scan utility
- Utility control statements for executing the Database Delete utility

When you use the two utilities, the Database Scan utility retrieves data and the Database 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
*
```

Note: You might need to include a SYSOUT DD card depending on the DFSORT DD cards used in your system.

Sample JCL for the Database Delete 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
```

Range of processing for the DBFUMDL0 utility

Sequential dependents (SDEPs) must be deleted, beginning with the oldest current SDEP, so that 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 (STOPRBA command) is not specified, default end processing is invoked. In the high-water-mark (HWM) CI, a permanent default end time-stamp segment is built. The delete starts with the oldest sequential dependent CI boundary in the area. The delete ends with the time stamp from the HWM CI default end segment, which is excluded from the delete.

The following sequence occurs 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 command 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 time stamp in the CI is used as a stop time stamp.
- For V5COMP, the CI is excluded and the time stamp 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 time stamp and location attributes.
- For V5COMP, the specified segment is the stop point and the time stamp is ignored.

When default end is used:

- All segments with a time stamp less than the HWM CI default end segment time stamp are included.
- For V5COMP, the Logical Begin is 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.

Chapter 7. DEDB Sequential Dependent Scan utility (DBFUMSC0)

Use the DEDB Sequential Dependent Scan utility (DBFUMSC0) online to scan and copy sequential dependent segments to a sequential data set and then process this data set offline using your own programs.

This utility runs in the DB utility type dependent region.

The DBFUMSC0 utility uses an exit routine. The sample exit routine delivered with IMS is the Data Entry Database Sequential Dependent Scan utility exit routine (DBFUMSE1).

With this utility you can:

- Specify a range of segments to be copied. If the segment 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.
- Specify an exit routine (other than the one provided by IMS) that can change the segment content and length.
- Specify a load module that 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 CI subset is processed without sorting.

The DBFUMSC0 utility can be stopped at the end of an area by using the IMS /STOP REGION command.

The DBFUMSC0 utility cannot be restarted. If you attempt to restart the DBFUMSC0 utility by specifying the restart (REST) parameter, the utility attempts an initial run from the beginning of the area defined by the restart parameter.

Subsections:

- "Restrictions" on page 64
- "Prerequisites" on page 64
- "Requirements" on page 64
- "Recommendations" on page 64
- "Input and output" on page 64
- "Return codes" on page 64
- "JCL specifications" on page 64

© Copyright IBM Corp. 1974, 2015

Restrictions

You must execute the DEDB Sequential Dependent Scan utility in a separate job step. The program cannot switch from one utility or one database to another within the same job step.

This utility fails if an in-doubt segment is encountered and the INDOUBT is not specified on a SYSIN control statement.

Data that contains CI sequential segments that are formatted for IMS are allocated to each IMS partner. The stopping location for the formatted 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.

You cannot restart the DBFUMSC0 utility.

The DBFUMSC0 utility will work on Fast Path (DEDB) databases only.

Utilities that alter databases cannot be run while the database is quiesced.

Prerequisites

Currently, no prerequisites are documented for the DBFUMSC0 utility.

Requirements

Currently, no requirements are documented for the DBFUMSC0 utility.

Recommendations

Currently, no recommendations are documented for the DBFUMSC0 utility.

Input and output

For input, the DBFUMSC0 utility uses a control data set that contains the input parameters supplied by DEDB utility commands. The control data set is pointed to by the SYSIN DD statement.

The DBFUMSC0 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.

Return codes

The DBFUMSC0 utility does not issue return codes.

JCL specifications

The DBFUMSC0 utility is executed as a standard z/OS job. The JCL specifications for the DBFUMSC0 utility include a JOB statement, the EXEC statement, and the

DD statements. Processing options for the DBFUMSC0 utility are specified by DEDB utility commands in a control data set that you allocate to the utility.

The following JCL statements are required:

- An EXEC statement
- DD statements that define inputs and outputs

The DBFUMSC0 utility can contain sortwork data definition JCL statements to provide sort work space. The sortwork DD statements can be dynamically allocated by the sort program. The SYSOUT DD statement used by the sort program for SYSPRINT can be dynamically allocated by the caller and passed to the sort program. The SORTSETUP *exit_routine_name* parameter statement can provide an exit routine to tailor requirements. The DBFUMSC0 utility default is to sort the sequential dependent segments.

EXEC statement

The EXEC statement executes the DEDB Sequential Dependent Scan utility. This statement can either specify the FPUTIL procedure that contains the required JCL, or it can be in the form:

```
PGM=DFSRRC00
```

DD statements

STEPLIB DD
Describes the library that contains the DBFUMSC0 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 DEDB utility commands that control the processing options of the DBFUMSC0 utility. Specify AREA XXXXXXXX as a SYSIN DD card to select the area to scan.

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.

Related concepts:

Guidelines for writing IMS exit routines (Exit Routines)

Related reference:

Data Entry Database Sequential Dependent Scan utility exit routine (DBFUMSE1) (Exit Routines)

FPUTIL procedure (System Definition)

Control statements for the DBFUMSC0 utility

The DBFUMSC0 utility uses DEDB utility commands to specify its processing options. You code the commands in the data set pointed to by the SYSIN DD statement.

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.

The required DEDB utility commands used by the DBFUMSC0 are AREA and TYPE.

ALLFMNEW (optional)
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.

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 (required)
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 - 8 characters. The first character must be alphabetic, and the rest of the characters are alphanumeric. Alphabetic characters include @, #, and $.

ERRORACTION (optional)
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 (optional)
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 - 8 characters. The first character must be alphabetic, and the rest of the characters are alphanumeric. Alphabetic characters include @, #, and $.

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

GO (optional)
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 (optional)
Specifies that RBAs of in-doubt segments are to be written to the SYSPRINT output data set.

NOSORT (optional)
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 (optional)
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 (optional)
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 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 (optional)
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 (optional)
Specifies a start time in hex format. You can use HEX time stamps to specify much more detail. The HEX time stamp of the newest SDEP SEGMENT can be obtained using the POS call.

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

```
STARTHEXT X'C5C55972CA04D000'    /* 16 hex digits required.  */
 Note: C5C55972CA04D000   =   2010.092 18:39:58.787.661.000
```

STARTHEXT and STOPHEXT use the same time stamp format and are much more detailed than STARTIME and STOPTIME.

STARTIME (optional)
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=required 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'2010.092 18:39:58 -00:00'
```

The STARTIME and STOPTIME use the same time stamp format and are detailed down to the second only.

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

STARTRBA **(optional)**
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.

STARTROOT **(optional)**
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 **(optional)**
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 - 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.

STOPHEXT (optional)
Specifies a stop time in hex format. You can use HEX time stamps to specify much more detail. The HEX time stamp of the newest SDEP SEGMENT can be obtained using the POS call.

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

```
STOPHEXT  X'C5C55A5DB883FE33'   /* 16 hex digits required.  */
Note:  C5C55A5DB883FE33   =   2010.092 18:44:05.131.327.875
```

STARTHEXT and STOPHEXT use the same time stamp format and are much more detailed than STARTIME and STOPTIME.

STOPRBA **(optional)**
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 **(optional)**
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 (optional)
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 - 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 (optional)
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=required offset to UTC which when added to the UTC gives local time. This stop 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>
STOPTIME=C'2010.092 18:40:05 -00:00'
```

The STARTIME and STOPTIME use the same time stamp format and are detailed down to the second only.

TYPE (required)
Specifies either the scan, delete, or reorganization utility as the type of run.

Attention: 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.

V5COMP (optional)
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 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.
- 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 time stamps across the CIs.

Examples for the DBFUMSC0 utility

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
```

Range of scans for the DBFUMSC0 utility

You can specify the range of range of sequential dependent segments to process.

There are three ways to specify the range of sequential dependent segments to process. You can specify the starting location by using the STARTRBA, STARTROOT, or STARTSEQ command. You can specify the stopping location by using the STOPRBA , STOPROOT, or STOPSEQ command.

If you do not specify a range, default end processing is invoked. In the high-water-mark (HWM) CI, a permanent default end time-stamp 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 time stamp 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 time stamp 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 time-stamp and location attributes.
- For V5COMP, the specified segment is the start point and the time stamp is ignored.

When default start is used:

- All segments with a time stamp not less than the logical begin time stamp (LBTS) are 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.

The following examples demonstrate the use of marker segments:

- 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 time stamp in the CI is used as a stop time stamp.
- For V5COMP, the CI is excluded and the time stamp is ignored.

When a STOPRBA segment boundary is specified:
- The specified segment acts like a committed marker segment for time-stamp and location attributes.
- For V5COMP, the specified segment is the stop point and the time stamp is ignored.

When default end is used:
- All segments with a time stamp less than the high-water-mark (HWM) CI default end segment time stamp 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.

Chapter 8. HALDB Partition Data Set Initialization utility (DFSUPNT0)

Use the HALDB Partition Data Set Initialization utility (DFSUPNT0) to initialize HALDB partitions.

The DFSUPNT0 utility normally initializes only partitions that are flagged in the RECON data set as needing to be initialized (PINIT=Y); however, you can have the DFSUPNT0 utility initialize partitions unconditionally, which initializes partitions even when they are recorded in the RECON data set as PINIT=N. Any data in a partition that is initialized unconditionally is lost.

You can specify either HALDB master database names, partition names, or both:

- If a HALDB master database name is specified, then all of the partitions in the database that are identified in the RECON data set as requiring partition initialization 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 partition record in the RECON data set.
- If a DFSOVRDS DD statement is specified, then all partitions in a HALDB database are unconditionally initialized.

The DFSUPNT0 utility is recommended for use with existing HALDB databases because the utility provides complete checking and validation of each partition by IMS. To initialize partitions when converting a database to HALDB, you can also use the Database Prereorganization utility (DFSURPR0).

You can specify a HALDB master database name on the EXEC parameter list submitted to the IMS Region Controller or as a statement in the SYSIN data set. If you use SYSIN statements you can specify multiple HALDB master database and partition names.

You can specify partition names only as statements in the SYSIN data set. You can specify multiple partition names.

The DFSUPNT0 utility can be executed as multiple job tasks. Specifying each partition in a separate DFSUPNT0 job and then running those jobs simultaneously can greatly improve the overall partition initialization process.

You can also run multiple jobs without specifying partition names, but this is not recommended. Each task attempts to initialize all required partitions found in RECON and serializes 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.

Subsections:

- “Restrictions” on page 76
- “Prerequisites” on page 76

© Copyright IBM Corp. 1974, 2015

- "Requirements"
- "Recommendations"
- "Input and output"
- "Return codes"
- "JCL specifications" on page 77

Restrictions

The DFSUPNT0 utility will work on full-function HALDB databases only.

Utilities that alter databases cannot be run while the database is quiesced.

Prerequisites

Before you run the DFSUPNT0 utility, you must define all data sets to z/OS that will by dynamically allocated by the DFSUPNT0 utility.

Requirements

To run the DFSUPNT0 utility you must satisfy various operational requirements.

The following requirements 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.
- 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.

Recommendations

The DFSUPNT0 utility is recommended for use with existing HALDB databases. The utility provides complete checking and validation of each partition by IMS. To initialize partitions when converting a database to HALDB, you can also use the Database Prereorganization utility (DFSURPR0).

To improve overall performance of the partition initialization process for a HALDB database, use separate DFSUPNT0 jobs to initialize each partition in a HALDB database and then run those jobs simultaneously.

Input and output

The primary inputs to the DFSUPNT0 utility are partitions requiring initialization. The partition data sets do not need to be allocated.

The primary outputs of the DFSUPNT0 utility are initialized partitions.

Return codes

The following return codes are provided at program termination:

Code	Meaning
0	Successful completion.

8 Processing error or the format of the DFSOVRDS DD statement is incorrect.

12 Invalid input or IMS control block.

16 Environment or user error.

32 Abend returned from attach of IMS.

99 Internal logic error.

JCL specifications

The DFSUPNT0 utility is executed as a standard z/OS job. The JCL specifications for the DFSUPNT0 utility include a JOB statement, the EXEC statement, and the DD statements. One or more utility control statements must be/can be included with the JCL statements.

The following JCL statements are required:

- A JOB statement that you define to meet the specifications of your installation.
- 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. Partition names are only allowed as SYSIN statements.

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

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

In the following example, a master database name HDODB1 is specified in the EXEC statement:

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

Without a SYSIN, the EXEC statements must be in the form shown in the following example:

```
//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 the form shown in the following example:

```
//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
/*
```

This JCL initialized all database partitions recorded in RECON as PINIT for HALDB master databases DBVHDJ05 and DB0HIDK5. If a partition name is specified for a HALDB master database, the partition 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.

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.

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

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.

The following figure is an example of DFS391I output to SYSPRINT:

Figure 17. DFS391I 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
```

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. You can initialize a HALDB database and all of its partitions by using this override option, even when the PARTITION INIT NEEDED flag in the RECON is not set to Y.

The only valid input for the DFSOVRDS DD statement is the character string INITALL or DD DUMMY. Any input other than INITALL or DD DUMMY causes the utility to issue a WTO message DFS1987I with reason code 19: THE DFSOVRDS DD CARD HAS AN INVALID INPUT.

The format of the DFSOVRDS DD statement is shown in the following figure.

Figure 18. DD statement for unconditional partition initialization

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

Attention: All previously existing partition data for all the HALDB partitions is lost. Any input other than the string INITALL will be ignored by the utility, as if DFSOVRDS DD is not specified.

Control statement for the DFSUPNT0 utility

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

Examples for the DFSUPNT0 utility

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

Subsections:

- "Initialize partitions"
- "Initialize all partitions for a list of master database names"

Initialize partitions

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.RESLIB.

```
//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
/*
```

Initialize all partitions for a list of master database names

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. The partition data sets are dynamically allocated using information from RECON. RECON data sets are dynamically allocated using MDA members found in the IMSVS.RESLIB.

```
//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 9. HALDB Partition Definition utility (%DFSHALDB)

Use the HALDB Partition Definition utility to register HALDB master databases with DBRC, and to add, modify, and delete HALDB partitions. You can use this utility instead of the DBRC commands that provide similar HALDB management functions.

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

- Register HALDB master databases with DBRC
- Add HALDB partitions to an existing HALDB database.
- 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.

The HALDB Partition Definition utility does not significantly increase contention for the RECON data set in online IMS systems. 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.

You can start the HALDB Partition Definition utility from within ISPF using the following startup command:

```
TSO %DFSHALDB
```

You can also start the HALDB Partition Definition utility by using the IMS Application Menu.

Subsections:

- "Restrictions"
- "Prerequisites" on page 82
- "Requirements" on page 82
- "Recommendations" on page 82
- "Input and output" on page 82
- "JCL specifications for interactive mode" on page 83
- "JCL specifications for batch import and export" on page 84
- "Return codes" on page 86

Restrictions

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

- The HALDB Partition Definition utility operates only on databases that are identified as HALDB databases in the DBDLIB member. If you change the DBD after the HALDB definitions are in place, you might not be able to manipulate

© Copyright IBM Corp. 1974, 2015

the definitions. For example, you cannot change the number of data set groups without deleting and redefining the HALDB partition definitions.
- The HALDB definition of a particular database cannot be updated by multiple users simultaneously. 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).

Prerequisites

Before you can use the %DFSHALDB utility to register a new HALDB master database with DBRC, you must first define the DBD for the master database by using the Database Description Generation (DBDGEN) utility.

Requirements

Currently, no requirements are documented for the %DFSHALDB utility.

Recommendations

Currently, no recommendations are documented for the %DFSHALDB utility.

Input and output

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 and export 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.

DSPXRUN EXPORT example output messages

Exporting database PARTDBA using JCL similar to that shown in Figure 22 on page 85 would result in the following output:

Figure 19. %DFSHALDB output message for export

```
DSPM142I Start export to MEM=PARTDBA in DSN='IBMUSER.HALDB.EXPORT'
         from DBN=PARTDBA
DSPM143I The export file contains partition PAAA
DSPM143I The export file contains partition PAAB
```

```
DSPM143I The export file contains partition PAAC
DSPM143I The export file contains partition PAAD
DSPM219I Table PARTDBA was created successfully to dataset
         'IBMUSER.HALDB.EXPORT'
```

DSPXRUN IMPORT example output messages

Importing database PARTDBA using JCL similar to that shown in Figure 22 on page 85 would result in the following output:

Figure 20. %DFSHALDB output message for import

```
DSPM083I Start Import to DBN=PARTDBA from MEM=PARTDBA in
         DSN='IBMUSER.HALDB.EXPORT' Options=2
DSPM085I Imports start at 07/07/09 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.
```

JCL specifications for interactive mode

The following figure shows a sample TSO logon procedure for running the %DFSHALDB utility in the interactive ISPF panel mode.

Figure 21. TSO logon procedure to run the %DFSHALDB utility in interactive mode

```
//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
```

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. In this concatenation also include the system ISPF table library.

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.

JCL specifications for batch import and export

The import and export functions of the HALDB Partition Definition utility can be performed in batch jobs. The JCL must include normal ISPF DD statements and start ISPF.

The JCL must include the following elements:

- 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

The output from the export of a HALDB is a member of a PDS. The information about the HALDB is saved in the form of an ISPF table. The ISPF table is used as input for the import process. The export and import can be done from the ISPF panels or from batch jobs.

The batch job executes the standard ISPF command, ISPSTART, which sets up the ISPF environment and then starts the DSPXRUN command. The DSPXRUN command identifies the HALDB database, the import file to use, and the processing options.

The batch import of a HALDB can be done by submitting a batch ISPF job similar to the job shown in the following figure.

To import a database using a batch job, submit a batch ISPF job similar to the job shown in the figure below. All ISPF DD names are required. ISPF is invoked in batch, so all ISPF DDNAMES are required.

Figure 22. Sample JCL for batch import

```
//*
//DSPXRUN  EXEC PGM=IKJEFT01,DYNAMNBR=50,REGION=6M
//STEPLIB  DD DISP=SHR,DSN=IMS.SDFSRESL    /* IMS.SDFSRESL */
//         DD DISP=SHR,DSN=ISP.SISPLOAD
//         DD DISP=SHR,DSN=ISP.SISPLPA
//SYSPROC  DD DISP=SHR,DSN=IMS.SDFSEXEC    /* IMS rexx execs     */
//         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,               /* dummy ISPF 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    /* IMS ISPF panels    */
//ISPSLIB  DD DISP=SHR,DSN=ISP.SISPSLIB    /* IMS ISPF skeletons */
//         DD DISP=SHR,DSN=ISP.SISPSLIB
//ISPMLIB  DD DISP=SHR,DSN=IMS.SDFSMLIB    /* IMS ISPF messages  */
//         DD DISP=SHR,DSN=ISP.SISPMENU
//ISPTLIB  DD DISP=SHR,DSN=ISP.SISPTENU    /* IMS ISPF tables    */
//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))
```

Using batch to export or import partition information

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. The HALDB Partition Definition utility requires its tables to be in this set of data sets. In this concatenation also include the system ISPF table library.

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 want to continue.

Return codes

The following 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 concepts:

Using IMS utilities with HALDB Online Reorganization (Database Administration)

HALDB partition names and numbers (Database Administration)

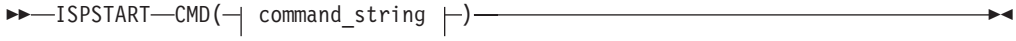
Control statements for the %DFSHALDB utility

The HALDB Partition Definition utility provides the batch DSPXRUN utility control statement. DSPXRUN supports the batch import and export of HALDB definitions. DSPXRUN is submitted as a parameter of the ISPF command ISPSTART, which supports the execution of ISPF code in general.

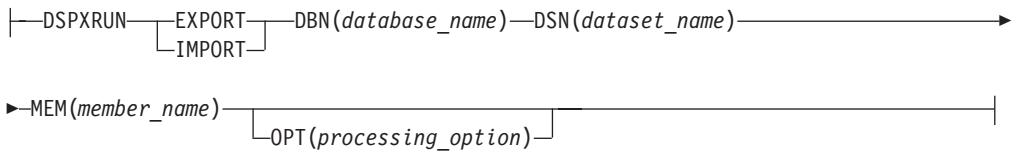
The utility control statements follow normal TSO command syntax and continuation rules:

- 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:



EXPORT
When you choose to export database information using a batch job, the information is stored in the partitioned data set that you specify. The information is saved as an ISPF table and so it must have the attributes of ISPTLIB data sets: record format = fixed block, record length = 80, and data set organization = PDS (or PDS/E).

IMPORT
When you choose to import database information using a batch job, the partition information is read from a partitioned data set that you specify. The partition information is defined to the RECON data sets.

database_name
The database name that was specified in the primary panel when the Export file was created by the HALDB Partition Definition utility.

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
The processing option field lets you determine what the Partition Definition utility does in the event that an error occurs when it processes a partition from the imported table. The Partition Definition utility records each partition it imports in RECON. If there are errors, you can choose to try the remaining partitions or to stop the process. The valid values are 1 or 2:

1 Stop on first error (prior imported partitions are retained)

2 Try all partitions

The OPT parameter is ignored during export processing.

Related tasks:

z/OS: Allocating optional table and file tailoring ISPF libraries

Examples for the %DFSHALDB utility

These examples show how to use the %DFSHALDB utility.

The examples assume that JCL similar to Figure 22 on page 85 is provided. The examples include the SYSTSIN statement as a point of reference.

Subsections:

- "Database definition export"
- "Prevent Import database definition with error"
- "Import independent HALDBs"

Database definition export

This example exports a database definition.

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

Prevent Import database definition with error

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)   +
                 )
 /*
```

Import independent HALDBs

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)  )
 /*
```

Running the %DFSHALDB utility

The HALDB Partition Definition utility is primarily executed as an interactive application; however, you can also run the import and export functions of the HALDB Partition Definition utility in batch.

When running as an interactive program, 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.

The HALDB Partition Definition utility is started from within ISPF. On the ISPF command line, type the following command:

```
TSO %DFSHALDB
```

The HALDB Partition Definition utility can also be started from the IMS Application Menu.

HALDB Partition Definition utility ISPF panels

The ISPF panels of the HALDB Partition Definition utility allow you to manage the partitions of an IMS HALDB.

To access the HALDB Partition Definition utility:

1. Log on to TSO.
2. Start ISPF.
3. From the ISPF command line, type: tso %dfshaldb and press Enter.

The utility consists of several panels and programs that perform various actions on the HALDB and its partitions.

Important: The Panel IDs are shown enclosed in parentheses in the caption of each panel image here. To display Panel IDs in the upper left corner of each of your panels, type panelid on the ISPF command line and press Enter.

Subsections:

- "Partitioned Databases panel"
- "Displaying the ISPF member list" on page 90
- "Opening HALDB partitions" on page 92
- "Defining data set group information" on page 101
- "Displaying the list of defined partitions" on page 103
- "Opening database information" on page 111
- "Deleting database information" on page 111
- "Exporting database information" on page 112
- "Importing database information" on page 112
- "Displaying the HALDB data set concatenation" on page 113
- "Selecting an IMS configuration" on page 114

Partitioned Databases panel

You define the HALDB that you want to manipulate on the Partitioned Databases panel. Specify the type of action to perform, for example, define, modify, or view. The succeeding panels guide you through the processes.

The Partitioned Databases panel has *point-and-shoot* text fields (in turquoise by default). To use the point-and-shoot fields, position the cursor on the text and press Enter.

The figure below provides space for you to enter a HALDB name, allowing HALDB to gather information about that HALDB. The information can be retrieved from DBDLIB or from the RECON data set, depending on the option you select and the current state of the partitions.

```
  Help
------------------------------------------------------------------------------

                           Partitioned Databases
Type a database name and choose an option.  Then press Enter.
To select a database from a list, type a filter (*) and press F4.
 Configuration  . . : DEFAULT
 Database name  . . . IVPDB1   +
 Option . . . . . . . __  1.  OPEN DATABASE partitions
                          2.  Open database information
                          3.  Delete database information
                          4.  Export database definitions
                          5.  Import database definitions
                          6.  Show IMS DDname concatenation
                          7.  Select IMS RECON / DBDLIB libraries
 To exit the application, press F3.
Command ===>
  F1=Help    F3=Exit  F4=Prompt
```

Figure 23. Partitioned Databases panel (DSPXPAA)

Configuration
The configuration is a name that you have specified that identifies a set of DBD libraries and a set of RECON data sets. If you already have the IMS DD statement allocated from the logon procedure and if you have the IMS.SDFSRESLs allocated to the STEPLIB DD statement, you do not need to use the **Configuration** option. If you do define and select a configuration, those data sets override the allocations from the logon procedure.

Database name
Enter up to 8 alphanumeric characters (the first character must be alphabetic). The HALDB name must be a member from a DBDLIB data set. DBDLIB data sets must be allocated under a DD name of IMS. The database name that you specify is remembered across ISPF sessions.

You can include an asterisk to indicate that you want a member list display. The asterisk can appear alone or as part of the name to limit the list that is displayed.

Important: When you include an asterisk as part of the member name, the concatenation for the IMS DD name may contain only up to four data sets. This is an ISPF restriction.

Option
A numeric value that indicates the type of processing to perform. The number corresponds to one of the actions in the list.

Displaying the ISPF member list

When you include an asterisk in the database name field, a member list for the members of the IMS DD name concatenation with a name that matches the filter is

displayed. A sample member list display is shown in the following figure.

```
  File  Help
------------------------------------------------------------------------------

 MEMBER LIST  IMSIVP81.DBDLIB                              Row 00001 of 00011
    Name                Size    TTR   Alias-of AC  AM  RM  ---- Attributes ---

 . DBFSAMD1           00000158 00013B          00  24  24
 . DBFSAMD2           000001A0 000143          00  24  24
 . DBFSAMD3           000006E0 00014B          00  24  24
 . DBFSAMD4           000002C8 000207          00  24  24
 . DI21PART           00000230 000133          00  24  24
 . IVPDB1             00000138 000103          00  24  24
 . IVPDB1I            00000138 00010B          00  24  24
 . IVPDB2             00000130 000113          00  24  24
 . IVPDB3             00000188 00011B          00  24  24
 . IVPDB4             00000110 000123          00  24  24
 . IVPDB5             000000B0 00012B          00  24  24
   **End**
 Command ====>                                              Scroll ===> CSR
  F1=Help    F3=Exit   F12=Cancel
```

Figure 24. ISPF member list display (DSPXPAM)

The member list originates from the PDS directories of the IMS concatenation. The members that are displayed can be HALDB or non-HALDB. The member list is a standard ISPF list so there is no IMS-specific information displayed.

From the member list, you can select the HALDB name to process by typing in the far-left column. If the name selected is not for a partitioned database, an error message is displayed. You can select a HALDB name with the slash (/) character and the File action to select the type of actions to perform. The same actions that are shown on the Partitioned Database panel (DSXPAA) are available here.

If you specify an option on the Partitioned Databases panel, you do not need to use the File Action bar; just press Enter. You can use the File Action bar to override the option that you specified an option on the Partitioned Databases panel.

The list of HALDBs in the Member List panel can be manipulated by using the File action bar.

Figure 25. File action bar choices

The options on the File Action bar allow you to perform the following actions:

- Create or change HALDB partitions.
- View or change HALDB information.
- Delete HALDB information.

- Export HALDB information.
- Import HALDB information.

Opening HALDB partitions

Before you can define the partitions for a HALDB, you must use the DBDGEN process to define the HALDB as a partitioned database.

The first time you choose a HALDB you must set values for the HALDB master. When you press Enter to continue, you set the defaults for the partitions. When you press Enter again, you define partitions using those defaults. You can modify each partition uniquely as they are created or you can modify them later from the list of partitions. Figure 29 on page 99 shows an example of the panel to specify the partition information.

After the initial set of partitions is defined (and whenever you select that HALDB again), you will see the Database Partitions display.

Important: Most of the information that is initially displayed on the panel in Figure 26 is extracted from the DBDLIB member. You can change the displayed information, but that information is saved in the RECON data sets, and is not saved in the DBDLIB member.

Each HALDB can support up to 1001 partitions.

```
 Help
-------------------------------------------------------------------------------

                      Partitioned Database Information
 Type the field values.  Then press Enter to continue.
 Database name . . . . . . . : IVPDB1
                     Master Database values
 Part. selection routine  . . . DFSIVD1
 RSR global service group . . . BKUPGRP1
 RSR tracking type  . . . . . . DBTRACK
 Share level  . . . . . . . . . 0
 Database organization  . . . : PHDAM
 Recoverable? . . . . . . . . . Yes
 Number of data set groups  . : 10
 Online Reorganization Capable: Yes
 To exit the application, press F3.
Command ===>
  F1=Help    F3=Exit    F12=Cancel
```

Figure 26. Partitioned Database Information (DSPXPOA)

The following are descriptions of the fields on the Partitioned Database information screen:

Database name
Enter 1 - 8 alphanumeric characters. This is the name you selected from the previous panel; it is the name of the HALDB that you are defining.

Part. selection routine
Enter 1 - 8 alphanumeric characters (the first character must be alphabetic). This is the name of the Partition Selection Exit Routine provided by you.

RSR global service group
Enter 1 - 8 alphanumeric characters (the first character must be alphabetic). This is an optional parameter that is used to specify the RSR global service group that the HALDB is to be assigned to.

RSR tracking type
This is an optional parameter that you use to specify the type of RSR tracking (shadowing) for a partition that is assigned to a global service group. The type, RCVTRACK or DBTRACK, cannot be specified without an RSR global service group having been defined for the HALDB master.

- DBTRACK- indicates HALDB readiness tracking is to be done.
- RCVTRACK- indicates recovery readiness tracking is to be done.

DBTRACK is the default.

Share level
0, 1, 2, or 3. Share level is an optional parameter. You can use share level to specify the level of HALDB data sharing with authorized subsystems.

Share level 0 is the default.

Database organization
This field indicates the type of HALDB organization, you can specify either: PSINDEX, PHIDAM, or PHDAM.

Recoverable?
Yes indicates that the HALDB is recoverable. No indicates that the HALDB is not recoverable. Yes is the default. If an RSR global service group is specified, the recoverable field must be Yes.

Number of data set groups
This is the number of data sets in the groups that contain data as specified in the DBDGEN.

Online reorganization capable
Yes specifies that this HALDB supports online reorganization. No specifies that this HALDB does not support online reorganization. These specifications are stored in the DBRC RECON data set.

The following figure shows the partition default information.

```
 Help
-------------------------------------------------------------------------------

                        Partition Default Information
 Type the field values.  Then press Enter to continue.
 Database name . . . . . : IVPDB1
                    Processing options
 Automatic definition  . . . . No
 Input dataset . . . . . . . . 'IMS.IVPDB1.KEYS'
 Use defaults for DS groups. . No
                    Defaults for partitions
 Partition name  . . . . . . . IVPD101
 Data set name prefix  . . . . IMS.DB01.FINANCE.YEAR2013.CURR  Randomizer
    Module name  . . . . . . . DD41DUP2
    Anchor . . . . . . . . . . 2
    High block number. . . . . 999
    Bytes  . . . . . . . . . . 2000
 Free Space
    Free block freq. factor. . 0
    Free space percentage. . . 0
                    Defaults for data set groups
 Block Size  . . . . . . . . . 8192
 DBRC options
    Max. image copies. . . . . 2
    Recovery period. . . . . . 0
    Recovery utility JCL . . . RECOVJCL
    Default JCL. . . . . . . . ________
    Image copy JCL . . . . . . ICJCL
    Online image copy JCL. . . OICJCL
    Receive JCL. . . . . . . . RECVJCL
    Reusable?  . . . . . . . . No
 To exit the application, press F3.
Command ===>
  F1=Help    F3=Exit   F6=Groups   F12=Cancel
```

Figure 27. Partition Default Information (DSPXPCA)

Important: The Randomizer section is present only if the HALDB is PHDAM. The Defaults for data set groups section is present only if there is only one data set group specified during DBDGEN. If there are multiple data set groups, use F6=Groups to display all data set groups using the dialog described in "Defining data set group information" on page 101.

The following are descriptions of the fields on the Partition Default Information screen:

Database name
This is the name that you selected from the previous panel (see Figure 23 on page 90), it is the name of the HALDB that you are defining.

Automatic definition
The value can be Yes or No. Specifying yes causes the partitions to be defined automatically based on your choices for partition name (that must include percent sign characters for placeholders.

Specifying No allows you to specify unique values for each partition.

Yes is the default.

Input data set
Provide the name of a z/OS data set. Specify a member name if it is a PDS. Each line of the data set must contain a partition selection string or the high key value to be used during partition definition.

Use defaults for DS groups
This value can be Yes or No. This option determines if all data set groups

are automatically set to the same defaults or if you are prompted to provide values for each group. It can be left blank if *automatic definition* is set to Yes.

Partition name
Enter 1 - 7 alphanumeric characters (the first character must be alphabetic). The Partition name is used as a prefix to the DDNAMEs of its data sets, and so it must be unique.

For automatic definitions, you must include percent signs (%) as placeholders for an alphanumeric sequence number (A - Z, 0 - 9).

Data set name prefix
Any alphanumeric name that is valid in JCL with a maximum length of 37 characters.

Module name
Enter 1 - 8 alphanumeric characters (the first character must be alphabetic). This is the name of the randomizing module. A randomizing module controls root segment placement in, or retrieval from, the PHDAM HALDB. This parameter is for PHDAM HALDBs only.

Anchor
1 - 3 numeric digits, with a range of 1 to 255. Specifies the number of root anchor points that are desired in each control interval or block in the root addressable area of a PHDAM HALDB. The anchor operand must be an unsigned decimal integer and must not exceed a value of 255. Typical values are from 1 to 5. This parameter is for PHDAM HALDBs only.

The default value is 1.

High block number
A numeric unsigned decimal integer value with a range of 0 to 2**24 - 1. This value specifies the maximum relative block number value that the user wants to allow a randomizing module to produce for this HALDB. This parameter is for PHDAM HALDBs only. This value determines the number of control intervals or blocks in the root addressable area of an PHDAM HALDB.

A high block number of zero means that no upper limit check is performed on the RBN created by the randomizing module. That is, it is all root addressable area.

Bytes A numeric unsigned decimal integer value with a range of 1 to 2**24 - 1. This value specifies the maximum number of bytes of a HALDB record that can be stored into the root addressable area in a series of inserts unbroken by a call to another HALDB record.

A value of 0 (zero) means that all bytes are addressable. It is equivalent to omitting the bytes parameter from the RMNAME keyword in the DBD macro statement in DBDGEN. This parameter is for PHDAM HALDBs only.

Free block freq. factor
A numeric unsigned decimal integer from 0 to 100, except 1. The free block frequency factor (fbff) specifies that every *nth* control interval or block in this data set group is left as free space during HALDB load or reorganization (where fbff=n). The range of fbff includes all integer values from 0 to 100 except fbff=1. The default value for fbff is 0.

Free space percentage
Two numeric unsigned decimal integer digits with a range from 0 to 99.

The *fspf* is the free space percentage factor. It specifies the minimum percentage of each control interval or block that is to be left as free space in this data set group.

The default value for *fspf* is 0.

Block size
A numeric unsigned even decimal integer with a range from 1 to 32000. *The block size value is used by OSAM only*. An initial value of 4096 is displayed. If the HALDB is not OSAM, the block size field is not displayed.

Max. image copies
A required parameter that you use to specify the number of image copies that DBRC maintains for the identified DBDS. The value must be an unsigned decimal integer from 2 to 255.

Recovery period
An optional parameter you use to specify the recovery period of the image copies for the specified DBDS.

Specify an unsigned decimal integer from 0 to 999 that represents the number of days that information about the image copies is kept in RECON. If you specify 0, there is no recovery period. 0 is the default.

Recovery utility JCL
Enter 1 - 8 alphanumeric characters (the first character must be alphabetic). This is an optional parameter that you use to specify the name of a member of a partitioned data set of skeletal JCL. When you issue the GENJCL.RECOV command, DBRC uses this member to generate the JCL to run the Database Recovery utility for the identified DBDS.

RECOVJCL is the default member name.

Default JCL
Enter 1 - 8 alphanumeric characters (the first character must be alphabetic). This is an optional parameter that you use to specify an implicit skeletal JCL default member for the DBDS. The specified member is used by the GENJCL.IC, GENJCL.OIC, and GENJCL.RECOV commands to resolve keywords that you have defined.

Image copy JCL
Enter 1 - 8 alphanumeric characters (the first character must be alphabetic). This is an optional parameter that you use to specify the name of a member of a partitioned data set that contains skeletal JCL. When you issue the GENJCL.IC command, DBRC uses this member to generate the JCL to run the Database Image Copy utility for the identified DBDS.

ICJCL is the default member name.

Online image copy JCL
Enter 1 - 8 alphanumeric characters (the first character must be alphabetic). This is an optional parameter that you use to specify the name of a member of a partitioned data set that contains skeletal JCL. DBRC uses this member when you issue the GENJCL.OIC command to generate the JCL to run the Online Database Image Copy utility for the identified DBDS.

OICJCL is the default member name.

Receive JCL
Enter 1 - 8 alphanumeric characters (the first character must be alphabetic).

This is an optional parameter that you use to specify the name of the skeletal JCL member that is used by the GENJCL.RECEIVE command.

RECVJCL is the default member name.

Reusable?

The value is either Yes or No. Specifies whether the Database Image Copy utility, or the Online Database Image Copy utility are to reuse previously defined image copy data sets.

No is the default value.

Automatic partition definition

In the Partition Default Information panel, you can set *Automatic definition* to yes and have your partitions defined without intervention. You must have previously created a data set and it must contain your partition selection strings. Specify the name of the data set in the *input data set* field of the panel.

Each line of the input data set must contain a partition selection string or the high key value to be used during partition definition. The file must contain only one value on each line of the file, with the value left-justified. The length of the string is determined by the last nonblank character. Each record must contain only one string.

In the partition name field, include percent signs (%) as placeholders for an alphanumeric sequence number (A - Z, 0 - 9). If you type a partition name like:

```
Partition name . . . . . IVPD1%%
```

The partitions are created in the following sequence:

```
IVPD1AA
IVPD1AB
IVPD1AC
.
.
IVPD1AZ
IVPD1A0
IVPD1A1
IVPD1A2
.
.
IVPD1A9
IVPD1BA
IVPD1BB
IVPD1BC
.
.
```

When you press Enter, as many partitions as you have key values in the input data set are automatically generated.

If you want to generate partition names that enable you to preserve your naming sequence when your database expands, you can specify a partition name like IVP1%%A. The partitions are then created in the following sequence:

```
IVP1AAA
IVP1ABA
.
.
IVP1AZA
IVP1A0A
IVP1A1A
```

```
.
.
IVP1A9A
IVP1BAA
.
.
```

When automatic definition is processing, a status panel is displayed (as shown in the following figure). This automatic definition status panel is updated as new partitions are defined.

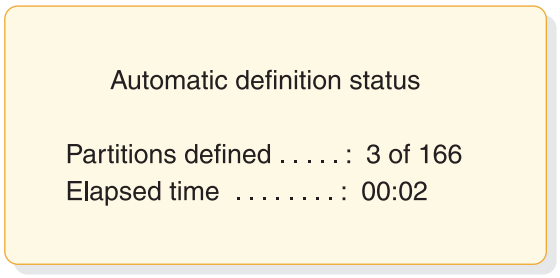


Figure 28. Automatic definition status

After automatic definition is complete, in the Database Partitions panel, you can see that the partition selection string is filled-in with information from your input data set.

Manual partition definition

On the Partition Default Information panel, you can set **Automatic definition** to No so that you can define the partitions serially. You can still use an input data set even though you set **Automatic definition** to No.

- If you specify an input data set, you must have previously created the data set and it must contain your partition selection strings. The partition selection string field in Change Partition panel (DSPXPPA) is primed from your input data set. For each partition, the partition selection string is filled-in from a record of the input data set. If you try to define more partitions than there are key values, the last key value from the input data set is displayed on the Change Partition panel and you will have to change it manually.
- If you do not specify an input data set to provide the partition high key values, the partition high key values can be added manually for each partition.
  - If you did not specify a partition selection exit, the partition high key values are required.
  - If you did specify a partition selection exit, the partition selection string values are optional.

After you set the defaults and press the enter key, the partition definition screen is displayed. You can modify the fields and press the enter key to define the partition. After you press the enter key, the partition is defined in RECON and the partition definition panel is displayed again so that you can define more partitions. The partition ID is incremented each time that a partition is defined. Press the cancel key (PF12) to prevent the displayed partition from being defined.

When you press PF12 to stop defining new partitions, the Partitioned Databases panel is displayed again. You can also choose to stop defining new partitions by

pressing F11=List; a list of defined partitions is displayed.

```
 Help
--------------------------------------------------------------------------------

                           Change Partition
 Type the field values.  Then press Enter.
 Database name . . . . . . . : IVPDB1
 Partition name  . . . . . . . IVPD101
 Partition ID. . . . . . . . : 1
 Data set name prefix. . . . . IMS.DB01.FINANCE.YEAR2013.CURR
 Partition Status. . . . . . . _______

 Partition Selection String
   +00 F2F0F0F3 4BF2F2F4 40F1F77A F2F57AF0 | 2013.224 17:25:0 |
   +10 F94BF6F3 F3F12432 00000000 00001020 | 9.6331.......... |
   +20 A840C1A5 85404040 40E28195 40D196A2 | y Ave    San Jos |
   +30 856B40C3 C14040F9 F5F1F4F1 00100020 | e, CA  95141.... |
   +40 00050000 40F0F34B F0F3F440 00000100 | .... 03.034 .... |
   +50 F1F8F0F0 C9C2D4E2 C5D9E540 40C9C2D4 | 1800IBMSERV  IBM |
   +60 40C39699 974B4040 F5F5F540 C2818993 |  Corp.  555 Bail |
   +70 A840C1A5 85404040 40E28195 40D196A2 | y Ave    San Jos |
   +80 856B40C3 C14040F9 F5F1F4F1 00403010 | e, CA  95141. .. |
   +90 00010500 40F0F34B F2F4F340 00324020 | .... 03.243 .. . |
   +A0 9201913C D2FE933D 913C1F66 4360A005 | k.j.K.l.j....-.. |
   +B0 3233A200 D996A281 6BD785A3 85996B40 | ..s.Rosa,Peter,  |
   +C0 000080D4 81A3A3F9 71C4C6F8 F1F4C6C2 | ...Matt9.DF814FB |
   +D0 9311913C F6F4F8F6 943C1F66 4360A005 | l.j.6486m....-.. |
   +E0 41E3453C 06000045 10110220 10416220 | .T.............. |
   +F0 FFFFF900 00004920 18007410 94000300 | ..9.........m... |

 Randomizer
    Module name  . . . . . . . DD41DUP2
    Anchor . . . . . . . . . . 2
    High block number. . . . . 999
    Bytes  . . . . . . . . . . 2000

 Free Space
    Free block freq. factor. . 0
    Free space percentage. . . 0

Attributes for data set group A
   Block Size  . . . . . . . . 8192


   DBRC options
      Max. image copies. . . . 2
      Recovery period. . . . . 0
      Recovery utility JCL . . RECOVJCL
      Default JCL. . . . . . . _______
      Image copy JCL . . . . . ICJCL
      Online image copy JCL. . OICJCL
      Receive JCL. . . . . . . RECVJCL
      Reusable?  . . . . . . . No
Command ===>
    F1=Help   F3=Exit   F5=String   F6=Groups   F12=Cancel
```

Figure 29. Change Partition (DSPXPPA)

Important: The Randomizer section is present only if the HALDB is PHDAM. The data set group attributes section is present only if there is only one data set group specified during DBDGEN. If there is more than one data set group, use F6=Groups to display all data set groups using the dialog that is described in "Defining data set group information" on page 101.

The following are descriptions of the fields on the Change Partition screen:

Partition ID

A numeric value between 1 and 32 767, but less than the current high partition ID value for this HALDB. The Partition Definition utility

generates the partition ID for you, regardless of whether you create your partitions manually or automatically. DBRC records this number in the RECON data set. Data set names include the partition ID of the partition to which they belong.

After an ID is assigned to a partition, you cannot change it.

Partition status

You can disable a partition by typing disable in the **Partition Status** field. Usually, you would disable a partition only prior to deleting it.

To enable a disabled partition, type enable in the **Partition Status** field.

Partition high key

The Partition High Key field allows you to specify the highest database record root key that a partition can contain. The partition high key is determined by your installation. IMS treats the partition high key as a hexadecimal value. You must enter a value in the Partition High Key field.

The length of the Partition High Key field is determined by the root key length you specify using the BYTES= parameter in the FIELD statement during DBD definition. If the length of the partition high key you enter is longer than the root key length, an error message displays and you must reduce the length of the partition high key. If the partition high key length is less than the defined root key length, the Partition Definition utility pads the high key value with hex 'FF's up to the defined root key length. The partition high key values must be unique for each partition within a HALDB.

The Partition High Key field consists of two sections: an editable section on the left that displays the partition high key in hexadecimal format and a view-only section on the right that displays the partition high key in alphanumeric format.

You can enter a hexadecimal value directly in the left section of the Partition High Key field. The Partition Definition utility displays the alphanumeric equivalent of this value in the right section of the Partition High Key field.

You can enter an alphanumeric value directly by using the ISPF editor. To access the ISPF editor, press F5 (If you have already entered something in the hexadecimal section, press F5 twice). After an alphanumeric value is entered, its hexadecimal equivalent is displayed in the left section of the Partition High Key field.

An alphanumeric value can consist of any character information. If the alphanumeric value contains non-display characters, you must identify these characters using hexadecimal notation. In the ISPF editor, a hexadecimal character string is enclosed by single quotation marks and either prefixed or followed with an x, for example, X'c1f201ffff'.

Partition selection string

The Change Partition panel displays the **Partition selection string** field only when you have specified a partition selection routine in the HALDB master definition. A partition selection routine uses the partition selection string in hexadecimal format to distribute records across the partitions in your HALDB.

Partition selection strings are 256 bytes long. If you enter a partition selection string that is less than 256 bytes in length, the Partition Definition utility fills the remaining bytes with X'00'.

The **Partition selection string** field consists of two sections: an editable section on the left that displays the partition selection string in hexadecimal format and a view-only section on the right that displays the partition selection string in alphanumeric format.

You can enter a hexadecimal value directly in the left section of the **Partition selection string** field. The Partition Definition utility displays the alphanumeric equivalent of this value in the right section of the Partition Selection String field.

You can enter the partition selection string in an alphanumeric format by using the ISPF editor. To access the ISPF editor, press F5 (If you have already entered something in the hexadecimal section, press F5 twice). After you enter an alphanumeric string, its hexadecimal equivalent is displayed in the left section of the **Partition selection string** field.

An alphanumeric string can consist of any character information. If an alphanumeric string contains non-display characters, you must identify these characters using hexadecimal notation. In the ISPF editor, a hexadecimal character string is enclosed by single quotation marks and either prefixed or followed with an x, for example: X'c1f201ffff'.

F5=String

F5 performs two functions: first, when new data is entered into the hexadecimal section of either the **Partition High Key** or the **Partition Selection String** field, F5 enters the data into the Partition Definition utility and displays the alphanumeric equivalent of the hexadecimal string in the right section of the field. Second, if there is no uncommitted data in the hexadecimal section, it displays the alphanumeric editor. The figure below is an example of the editor panel that is displayed for the **Partition Selection String** field.

```
EDIT       Partition Selection String
  Database name  . . . . : IVPDB1
  Partition name . . . . : IVPD101
****** ***************************** Top of Data **********
=COLS> ----+----1----+----2----+----3----+----4----+----5---

000001 '546787789af'x
****** **************************** Bottom of Data ********
Command ===>
    F1=Help   F3=Exit
```

Figure 30. Selection String editor (DSPXPKE)

F6=Groups

Pressing F6 displays the Data set group dialog.

F11=List

Pressing F11displays the Database partitions panel.

If your definition of the HALDB from DBDLIB allows only one data set group, the **Attributes for data set group A** section is displayed. If multiple groups are allowed, a reminder to press PF6 to work with the groups is displayed.

Defining data set group information

You can define data set group information by pressing F6 on the Change Partition panel. This section describes how to define the data set group information.

If you have multiple data set groups defined for your HALDB and you do not use automatic definition, use the data set group list.

From the data set groups list, you can change the attributes for each member by typing over the values in the *list* column. There is a special row in the list that allows you to make changes to an entire column of the list; the *all* row. When you type a value in the *all* row and press Enter, the value you typed is propagated to all of the members of the groups. After your changes are made, the *all* row is blanked out.

Important: Press F9 to save your changes and then press F12 to return to the previous panel.

The list contains an *action* column. The only action that is allowed is to display all information for a particular group. Select the group by typing a slash (/) in the *Act* column. The following figure shows how to modify the values by typing over the existing data and pressing enter.

```
  Help
------------------------------------------------------------------------------

                           Change Dataset Groups          Row 1 to 11 of 11
Select an item by pressing a '/' on the desired line then press Enter.
Database name . . . . . . . . : IVPDB1
Partition name  . . . . . . . : IVPD101
Partition ID. . . . . . . . . : 1
Data set name prefix. . . . . : IMS.DB01.FINANCE.YEAR2013.CURR
             Block      Max Image  Recovery   Recovery   Default
Act  Group   Size       Copies     Period     Util. JCL  JCL
___  All     _____      __         ___        ________   ________
___  A       8192       2          0          RECOVJCL   ________
___  B       8192       2          0          RECOVJCL   ________
___  C       8192       2          0          RECOVJCL   ________
___  D       8192       2          0          RECOVJCL   ________
___  E       8192       2          0          RECOVJCL   ________
___  F       8192       2          0          RECOVJCL   ________
___  G       8192       2          0          RECOVJCL   ________
___  H       8192       2          0          RECOVJCL   ________
___  I       8192       2          0          RECOVJCL   ________
___  J       8192       2          0          RECOVJCL   ________
Command ===>
 F1=Help  F3=Exit  F7=Backward  F8=Forward  F9=Save   F11=Right  F12=Cancel
```

Figure 31. Change Data Set Groups, part 1 (DSPXPGA)

```
  Help
-------------------------------------------------------------------------------

                             Change Dataset Groups          Row 1 to 11 of 11
Select an item by pressing a '/' on the desired line then press Enter.
Database name . . . . . . . : IVPDB1
Partition name  . . . . . . : IVPD101
Partition ID. . . . . . . . : 1
Data set name prefix. . . . : IMS.DB01.FINANCE.YEAR2013.CURR
             Image      On. Image  Receive
Act  Group   Copy JCL   Copy JCL   JCL        Reusable?
___  All     ________   ________   ________   ___
___  A       ICJCL      OICJCL     RECVJCL    No
___  B       ICJCL      OICJCL     RECVJCL    No
___  C       ICJCL      OICJCL     RECVJCL    No
___  D       ICJCL      OICJCL     RECVJCL    No
___  E       ICJCL      OICJCL     RECVJCL    No
___  F       ICJCL      OICJCL     RECVJCL    No
___  G       ICJCL      OICJCL     RECVJCL    No
___  H       ICJCL      OICJCL     RECVJCL    No
___  I       ICJCL      OICJCL     RECVJCL    No
___  J       ICJCL      OICJCL     RECVJCL    No
 Command ===>
 F1=Help  F3=Exit  F7=Backward  F8=Forward  F9=Save   F11=Right  F12=Cancel
```

Figure 32. Change Data Set Groups, part 2 (DSPXPGB)

```
  Help
-------------------------------------------------------------------------------

                             Change a Dataset Group
  Enter values, then press Enter.
  Attributes for data set group B
    Block size  . . . . . . . . . 8192


    DBRC options
       Max. image copies  . . . . 2
       Recovery period  . . . . . 0
       Recovery utility JCL . . . RECOVJCL
       Default JCL. . . . . . . . ________
       Image copy JCL . . . . . . ICJCL
       Online image copy JCL  . . OICJCL
       Receive JCL  . . . . . . . RECVJCL
       Reusable?  . . . . . . . . No
Command ===>
  F1=Help   F3=Exit   F12=Cancel
```

Figure 33. Change a Data Set Group (DSPXPGD)

Displaying the list of defined partitions

When you choose *Open database partitions* from the Partitioned Databases panel, the Database Partitions list is displayed. The list is displayed immediately if there are already partitions defined for the HALDB, or it is displayed after you define partitions for HALDBs that do not already have partitions. The following figure is an example of the Database Partitions list. The list is displayed as a table that you can scroll up and down in.

```
  File  Edit  View  Help
-------------------------------------------------------------------------------

                            Database Partitions           Row 1 to 15 of 166
Select an item by pressing a '/' on the desired line then press Enter.
Database name  . . . . . : IVPDB1
Act   Name      Id        Data set name prefix                        Status
___   IVPD101          1  IMS.DB01.FINANCE.YE2013
___   IVPD102          2  IMS.DB01.PAYROLL.YE2013
___   IVPD103          3  IMS.DB01.PAYROLL.YE2013
___   IVPD104          4  IMS.DB01.PAYROLL.YE2013
___   IVPD105          5  IMS.DB01.PAYROLL.YE2013
___   IVPD106          6  IMS.DB01.PAYROLL.YE2013
___   IVPD107          7  IMS.AB01.PAYROLL.YE2013
___   IVPD108          8  IMS.DB01.PAYROLL.YE2013
___   IVPD109          9  IMS.DB01.PAYROLL.YE2013
___   IVPD110         10  IMS.AB01.PAYROLL.YE2013                  Disabled
___   IVPD111         11  IMS.DB01.PAYROLL.YE2013
___   IVPD112         12  IMS.DB01.PAYROLL.YE2013
___   IVPD113         13  IMS.TP01.PAYROLL.YE2013
___   IVPD114         14  IMS.DB01.PAYROLL.YE2013
___   IVPD115         15  IMS.DB01.FINANCE.YE2013
      Command ===>
 F1=Help    F3=Exit  F7=Backward   F8=Forward  F11=Right
```

Figure 34. Database Partitions panel, sorted by partition ID (DSPXPLA)

The Database Partitions list panel has the HALDB name at the top and table information below. Descriptions of the table columns are listed below.

Act This is the line command input field where you can invoke commands such as open, copy, and the other commands.

Name The name column contains the partition name that was provided during the definition of the partition. This is the initial sort sequence.

Id This is the partition ID number. The number does not have to be sequential.

Data set name prefix
The data set name prefix contains the name of the data set that was provided during the definition of the partition.

Status A partition can be disabled by selecting a partition and typing "disable" in the Partition status field of the Change Partition panel. When a partition is disabled, "Disabled" appears in the Status column for that partition in the Database Partitions panel. For enabled partitions, the column remains blank.

From the Database Partitions list panel, you can work with individual partitions. To use the File Action bar, type a slash (/) in the *Act* line command column for the partition you want to work with, then put the cursor on the *File* action bar choice and press Enter. Select the action that you want to perform by typing the number or by positioning the cursor on the choice then pressing enter again.

You can invoke the Database Partitions panel to show the values by pressing your PF11 key.

```
  File  Edit  View  Help
-------------------------------------------------------------------------------

                             Database Partitions            Row 1 to 4 of 166
Select an item by pressing a '/' on the desired line then press Enter.
Database name  . . . . . : IVPDB1
Act   Name
             Partition Selection String
____  IVPD001
         +00 F2F0F0F3 4BF2F2F4 40F1F77A F2F57AF0 | 2013.224 17:25:0 |
         +10 F94BF6F3 F3F12432 00000000 00001020 | 9.6331.......... |
         +20 A840C1A5 85404040 40E28195 40D196A2 | y Ave    San Jos |
         +30 856B40C3 C14040F9 F5F1F4F1 00100020 | e, CA  95141.... |
         +40 00050000 40F0F34B F0F3F440 00000100 | .... 03.034 .... |
         +50 F1F8F0F0 C9C2D4E2 C5D9E540 40C9C2D4 | 1800IBMSERV  IBM |
         +60 40C39699 974B4040 F5F5F540 C2818993 |  Corp.  555 Bail |
         +70 A840C1A5 85404040 40E28195 40D196A2 | y Ave    San Jos |
         +80 856B40C3 C14040F9 F5F1F4F1 00403010 | e, CA  95141. .. |
         +90 00010500 40F0F34B F2F4F340 00324020 | .... 03.243 .. . |
         +A0 9201913C D2FE933D 913C1F66 4360A005 | k.j.K.l.j....-.. |
         +B0 3233A200 D996A281 6BD785A3 85996B40 | ..s.Rosa,Peter,  |
         +C0 000080D4 81A3A3F9 71C4C6F8 F1F4C6C2 | ...Matt9.DF814FB |
         +D0 9311913C F6F4F8F6 943C1F66 4360A005 | l.j.6486m....-.. |
         +E0 41E3453C 06000045 10110220 10416220 | .T.............. |
         +F0 FFFFF900 00004920 18007410 94000300 | ..9.........m... |
____  IVPD002
         +00 F2F0F0F3 4BF2F2F4 40F1F87A F1F27AF0 | 2013.224 18:12:0 |
         +10 F94BF6F3 F3F12432 00000000 00001020 | 9.6331.......... |
Command ===>
 F1=Help    F3=Exit  F7=Backward   F8=Forward   F11=Right
```

Figure 35. Database Partitions panel, sorted by key (DSPXPLB)

The Database Partitions list panel has the HALDB name at the top and table information below. The columns include the following:

Act This is the line command input field where you can invoke commands such as open, copy, and the other commands.

Name The name column contains the partition name that was provided during the definition of the partition. This is the initial sort sequence.

Partition selection string
The partition selection string is used by the partition selection routine.

You can invoke the Database Partitions panel to show the Randomizer values by pressing your PF11 key.

```
    File  Edit  View  Help
-------------------------------------------------------------------------------

                            Database Partitions           Row 1 to 15 of 166
Select an item by pressing a '/' on the desired line then press Enter.
Database name  . . . . . : IVPDB1
                ------------ Randomizer --------------    - Free Space -
Act   Name      Module     Anchor  High block  Bytes         FBFF   FSPF
___   IVPD101   DD41DUP2   2       999         2000          0      0
___   IVPD102   DD41DUP2   2       999         2000          0      0
___   IVPD103   DD41DUP2   2       999         2000          0      0
___   IVPD104   DD41DUP2   2       999         2000          0      0
___   IVPD105   DD41DUP2   2       999         2000          0      0
___   IVPD106   DD41DUP2   2       999         2000          0      0
___   IVPD107   DD41DUP2   2       999         2000          0      0
___   IVPD108   DD41DUP2   2       999         2000          0      0
___   IVPD109   DD41DUP2   2       999         2000          0      0
___   IVPD110   DD41DUP2   2       999         2000          0      0
___   IVPD111   DD41DUP2   2       999         2000          0      0
___   IVPD112   DD41DUP2   2       999         2000          0      0
___   IVPD113   DD41DUP2   2       999         2000          0      0
___   IVPD114   DD41DUP2   2       999         2000          0      0
___   IVPD115   DD41DUP2   2       999         2000          0      0
 Command ===>
 F1=Help    F3=Exit  F7=Backward   F8=Forward  F11=Right
```

Figure 36. Database Partitions panel, sorted by name (DSPXPLC)

Important: The Randomizer section is present only if the HALDB is PHDAM.

The Database Partitions list panel has the HALDB name at the top and table information below. The columns include the following:

Act This is the line command input field where you can invoke commands such as open, copy, and the other commands.

Name The name column contains the partition name that is provided during the definition of the partition. This is the initial sort sequence.

Module
The module column contains the module name of the randomizing module.

Anchor
The anchor column contains the number of root anchor points.

High block
The high block column contains the high block number.

Bytes The bytes field is described in Figure 27 on page 94.

FBFF The FBFF column contains the free block frequency factor.

FSPF The FSPF column contains the free space percentage factor.

To use line commands, type the command in the *Act* column to the right of the partition you want to use. You can type multiple line commands (only one per partition, though) on the Database Partitions panel: the commands are executed serially starting from the top.

The partition list line commands

Line commands allow you to perform the following actions:

Delete a partition
Type a D in the line command field and press Enter. A delete confirmation

panel is displayed. Type a 1 in the option field and press Enter to confirm the delete or press the cancel key to cancel the delete.

If you are deleting several partitions at once, and want to accept all of the deletes, you can type a 2 in the option field. It is reset to blank each time the partition list is displayed.

Copy a partition
Type a C in the line command field to define a new partition using the attributes of the selected partition. The partition name must be unique. You change the partition information using the Change Partition panel.

Open a partition
Type an O in the line command field to open a partition. You can then change partition information. The partition name and ID cannot be changed. press Enter to commit or press the cancel key to discard your changes. You change the partition information using the Change Partition panel.

Print partition information
Type a P in the line command field to print partition information for the selected partition. The information will not be routed to a printer immediately; instead it is added to the ISPF list data set.

The partition list action bar

The list of partitions in the Database Partitions panel can be manipulated with line commands or by using the File action bar choices, as shown in the figure below.

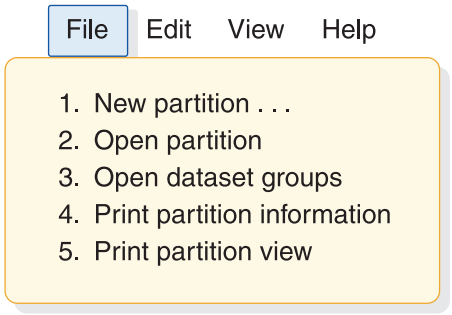


Figure 37. File action bar choices

New partition
You can create new partitions using the same panels that you used when you initially created partitions.

Open partition
You can open the selected partitions and modify them as needed.

Open data set groups
You can manipulate the data set group members using the panels that are described in "Defining data set group information" on page 101.

Print partition information
Information about the selected partitions is written to the ISPF list data set.

Print partition view
The information in the currently displayed view is written to the ISPF list data set.

The list of partitions in the Database Partitions can be sorted in various ways using the Edit action bar choice.

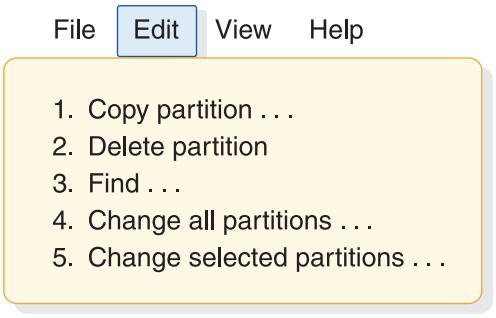


Figure 38. Edit action bar choices

Copy partition...
Type a slash (/) in the line command field and use the Edit - Copy partition pull-down panel to define a new partition using the attributes of the selected partition. The partition name and the ID must be unique.

The Change Partition panel is displayed, and you can create new partitions serially. The values shown in the panel are filled-in using the attributes of the selected partition.

Delete partition
Type a slash (/) in the line command field and use the Edit - Delete a partition pull-down panel to delete partitions. A delete confirmation panel is displayed. You can press Enter to confirm delete or press the cancel key to ignore the delete.

Find... You can search the partition list for a selected character string. Only simple character values can be specified. The cursor is placed on the partition that contains the search value.

The search string is not case sensitive. It searches on any field, not just the currently displayed fields on the Database Partitions panels, as show in the following figure.

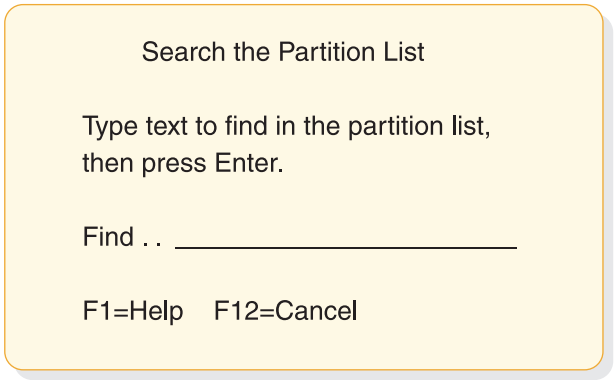


Figure 39. Searching the partition list

Change all partitions...
Use the Edit - Change all partitions pull-down panel to change individual fields for all of the partitions in the HALDB. The partition name and the ID cannot be changed.

Change selected partitions...
Type a slash (/) in the line command field to change a partition and use the Edit - Change selected partitions pull-down panel to change individual fields for the selected partitions. The partition name and the ID cannot be changed. Only the selected partitions are changed.

The list of partitions in the Database Partitions can be sorted in various ways by using the View action bar choice, which is shown in the following figure..

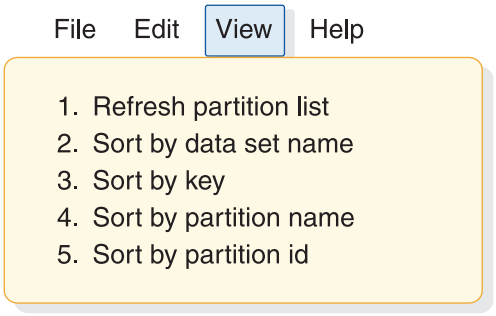


Figure 40. View action bar choices

Help information is available using the Help action bar choice.

Change all partitions

The Change Partition panel (DSPXPPB) is an example of the Change Partition panel. The entry fields are blank. Make changes only to the fields that you want to change. The field changes are applied to all of the partitions.

Important: The same process is used for Change selected partitions except that the changes are only applied to the partitions selected from the list with a slash (/).

If you want to change a character field to blanks, type a single slash (/) character so that it is the only character in the field.

```
  Help
-------------------------------------------------------------------------------

                          Change Partition
Press Enter to continue.
Database name.  . . . . . : IVPDB1
Partition name  . . . . . :
Partition ID  . . . . . :
Data set name prefix  . . . . _____________________________________
Status. . . . . . . . . . . . _______
Partition Selection String
  +00 ________ ________ ________ ________ |                  |
  +10 ________ ________ ________ ________ |                  |
  +20 ________ ________ ________ ________ |                  |
  +30 ________ ________ ________ ________ |                  |
  +40 ________ ________ ________ ________ |                  |
  +50 ________ ________ ________ ________ |                  |
  +60 ________ ________ ________ ________ |                  |
  +70 ________ ________ ________ ________ |                  |
  +80 ________ ________ ________ ________ |                  |
  +90 ________ ________ ________ ________ |                  |
  +A0 ________ ________ ________ ________ |                  |
  +B0 ________ ________ ________ ________ |                  |
  +C0 ________ ________ ________ ________ |                  |
  +D0 ________ ________ ________ ________ |                  |
  +E0 ________ ________ ________ ________ |                  |
  +F0 ________ ________ ________ ________ |                  |
Randomizer
   Module name  . . . . . DD41MOD3
   Anchor . . . . . ___
   High block number. . . . . ________
   Bytes  . . . . . ________
Free Space
   Free block freq. factor. . ___
   Free space percentage. . . __
Command ===>
   F1=Help  F3=Exit  F5=String  F6=Groups  F12=Cancel
```

Figure 41. Change Partition panel (DSPXPPB)

Important:

- The Randomizer section is present only if the HALDB is PHDAM.
- The data set groups section is present only if there is only one data set group specified during DBDGEN. If there is more than one data set group, use F6=Groups to display all data set groups using the dialog.

The following figure shows the Change Dataset Groups panel.

```
  Help
-------------------------------------------------------------------------------

                              Change Dataset Groups        Row 1 to 10 of 10
Select an item by pressing a '/' on the desired line then press Enter.
Database name . . . . . . . : IVPDB1
Partition name  . . . . . . : IVPD101
Partition ID. . . . . . . . : 1
Data set name prefix. . . . : IMS.DB01.FINANCE.YEAR2013.CURR
             Block      Max Image  Recovery   Recovery   Default
Act  Group   Size       Copies     Period     Util. JCL  JCL
___  All     _____      __         ___        ________   ________
___  A       _____      __         ___        ________   ________
___  B       _____      __         ___        ________   ________
___  C       _____      __         ___        ________   ________
___  D       _____      __         ___        ________   ________
___  E       _____      __         ___        ________   ________
___  F       _____      __         ___        ________   ________
___  G       _____      __         ___        ________   ________
___  H       _____      __         ___        ________   ________
___  I       _____      __         ___        ________   ________
___  J       _____      __         ___        ________   ________
Command ===>
 F1=Help  F3=Exit  F7=Backward  F8=Forward  F9=Save  F11=Right  F12=Cancel
```

Figure 42. Change Data Set Groups, part 1 (DSPXPGA)

Opening database information

When you choose Open database information from the Partitioned Databases panel, you are shown information about the HALDB which was saved when you first defined partitions for the HALDB.

```
  Help
-------------------------------------------------------------------------------

                      Partitioned Database Information
 Type the field values.  Then press Enter to continue.
 Database name . . . . . . . : IVPDB1
                     Master Database values
 Part. selection routine  . . . DFSIVD1
 RSR global service group . . .
 RSR tracking type  . . . . . .
 Share level  . . . . . . . . . 0
 Database organization  . . . : PHDAM
 Recoverable? . . . . . . . . . Yes
 Number of data set groups  . : 1
 Online Reorganization Capable: Yes
 To exit the application, press F3.
Command ===>
  F1=Help    F3=Exit   F12=Cancel
```

Figure 43. Partitioned Database Information (DSPXPOA)

You can modify the fields and press Enter to change the values in RECON. If you press cancel or exit, any changes you entered on this panel are discarded.

Deleting database information

When you choose to delete database information from the Partitioned Databases panel, you are presented with the Delete a Database panel. You must type a slash (/) character and press Enter to confirm the delete. When you confirm it, the information about the HALDB and about all of its partitions is deleted from RECON.

There is no way to undo the delete. You might want to perform an export prior to deleting a HALDB from RECON.

```
 Help
--------------------------------------------------------------------------------

                        Delete Database Information
 Type '/' to confirm the delete of the database information from RECON.
 Then press Enter.
 Database name . . . . . : IVPDB1
 Confirm database delete . __
Command ===>
  F1=Help    F3=Exit   F12=Cancel
```

Figure 44. Delete a Database (DSPXPDA)

Exporting database information

When you choose to Export database information from the Partitioned Databases panel, the information is stored in the partitioned data set that you specify. It is saved as an ISPF table and so must have the following ISPTLIB data set attributes:

- A fixed block record format
- A record length of 80
- A PDS (or PDS/E) data set organization

The following figure shows the Export a Database panel.

```
 Help
--------------------------------------------------------------------------------

                         Export a Database
 Type a data set name.  Then press Enter.
 Database Name  . . . . . : IVPDB1
 Output dataset name. . . . . 'TEST.RSR.PARTS'
 Output member name . . . . . IVPDB1
  To exit the application, press F3.
Command ===>
  F1=Help    F12=Cancel
```

Figure 45. Export a Database (DSPXPEA)

Field Description

Database name
The HALDB name that was specified in the primary panel.

Output data set name
The output data set name is the name of the data set that will contain the partition information.

You can also export HALDB database definitions by using a batch job and the Partition Definition utility control statement DSPXRUN.

Importing database information

When you choose to Import database information from the Partitioned Databases panel you can specify the name of the PDS or PDS/E that contains the information.

Important: Only an exported ISPF table can be used for importing database definitions. The output from the export of a HALDB is a member of a PDS. The information about the HALDB is saved in the form of an ISPF table. The ISPF table becomes input for the import process.

The import can be performed from the HALDB Partition Definition utility or a batch job.

After you press Enter, the table is read and each partition is defined.

The following figure shows the Import a Database panel.

```
 Help
--------------------------------------------------------------------------------

                         Import a Database
 Type a dataset name.  Then press Enter.
 Database name . . . . . : IVPDB1
 Input dataset name. . . . 'PROD.RSR.PARTS'
 Input member name . . . . IVPDB1
 Processing option . . . . __ 1. Stop on first error
                              2. Try all partitions
Command ===>
  F1=Help    F3=Exit   F12=Cancel
```

Figure 46. Import a Database (DSPXPIA)

Database name
The HALDB name that was specified on the primary panel.

Input 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 partitioned.

Input 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.

You can also import HALDB database definitions by using a batch job and the Partition Definition utility control statement DSPXRUN.

Displaying the HALDB data set concatenation

You can look at the concatenation of data sets that are allocated to the IMS ddname. The data sets are displayed using the ISRDDN command that is part of the ISPF product. The STEPLIB concatenation is not modified.

```
                        Current Data Set Allocations            Line 1 of 2
 Volume   Disposition Act DDname   Data Set Name   List Actions: B E V F C I Q
 SYS151   SHR,KEEP   > _  IMS      IMSIVP91.DBDLIB
 SYS335   SHR,KEEP   > _           IMS91.SANJOSE.DBDLIB
------------------------- End of Allocation List -----------------------------

Command ===>                                                Scroll ===> CSR
```

Figure 47. The IMS concatenation (ISRDDN)

When you specify a generic database name and use options 1 through 5 from the DFSHALDB panel, the viewing IMS ddname concatenation option works only if you use 4 or fewer DBD data sets. If you specify option 7, the data sets concatenated to the IMS ddname always display.

Use the help (F1) information that is provided by ISRDDN and ISPF to learn more about the ISRDDN utility. When you exit the ISRDDN utility, you return to the HALDB Partition Definition utility panels.

Selecting an IMS configuration

The HALDB Partition Definition utility menu contains an option to define IMS configurations. An IMS configuration is a set of DBDLIB and RECON data sets. You specify the configuration name.

If you have the IMS ddname allocated from the logon procedure and the IMS.SDFSRESL libraries allocated to the STEPLIB ddname, do not use the configuration option. If you define and select a configuration, those data sets override the allocations from the logon procedure.

You should use the same DBDLIB and RECON data sets that IMS will use to access the database. You can specify one data set for RECON1, RECON2, RECON3, and up to 10 DBDLIB data sets for the IMS ddname. You can control the RECON data sets in a configuration.

The IMS ddname includes the data sets that contain the DBDLIB members. The RECON / DBDLIB Configurations panels reallocate the IMS DD name.

The STEPLIB allocation contains the RECON1, RECON2, and RECON3 members that name the actual RECON data sets. IMS uses those members to determine which RECON data sets to use. As an alternative to using a STEPLIB, you can use the TSOLIB command to change the search order that TSO/E uses to find commands and programs. The RECON / DBDLIB Configurations panel reallocates the IMS ddname and allocates RECON1, RECON2, and RECON3 ddnames to explicitly specify the RECON data sets. The STEPLIB concatenation is not modified.

If you delete a configuration only, the configuration is deleted from the list, but the data sets that are named in the configuration are not deleted.

The following figure shows the RECON/DBDLIB Configurations panel.

```
                      RECON / DBDLIB Configurations          Row 1 to 5 of 5
To create a new configuration, fill in the first line and press Enter.
Select a default by type '/' on the Act column then press Enter.
You can use 'O' to open or 'D' to delete a configuration.
Act   Current  Name       Description

___              _______    ____________________________________________
___   *          SDFSRESL   IMS datasets
___              TESTM      Test IMS for Matt
___              TESTP      Test IMS for Peter
___              TEST1      Test IMS
*************************** Bottom of data ****************************
Command ===>
F1=Help     F3=Exit     F7=Up       F8=Down     F12=Cancel
```

Figure 48. User configurations (DSPXPMB)

A list of configurations can be maintained when you select option 7 from the Partitioned Databases panel. The list is initially empty and it can be added-to by filling in the blank line. The active configuration is identified by an asterisk (*) in the Current column.

Rows from the list can be deleted by using a line command of d. Only the configuration is deleted from the list. The data sets that are named in the configuration are not deleted.

The data sets named in the configuration are set or changed by using a line command of o for open.

The following figure shows the Configurations Details panel.

```
                           Configuration Details
Type in values in the fields and press Enter to continue.
Configuration name . . . . TEST1
Description  . . . . . . . Test IMSRECON dataset names
   RECON1 dataset  . . . . 'TEST.PARTS.RECON1'
   RECON2 dataset  . . . . 'TEST.PARTS.RECON2'
   RECON3 dataset  . . . . ________________________________
DBDLIB dataset names
   DBDLIB dataset 1  . . . 'TEST.PARTS.DBDLIB'
   DBDLIB dataset 2  . . . ________________________________
   DBDLIB dataset 3  . . . ________________________________
   DBDLIB dataset 4  . . . ________________________________
   DBDLIB dataset 5  . . . ________________________________
   DBDLIB dataset 6  . . . ________________________________
   DBDLIB dataset 7  . . . ________________________________
   DBDLIB dataset 8  . . . ________________________________
   DBDLIB dataset 9  . . . ________________________________
   DBDLIB dataset 10 . . . ________________________________
Command ===>
F13=Help     F15=Exit     F19=Up       F20=Down     F22=Actions
```

Figure 49. Configuration Details panel (DSPXPMC)

The RECON data sets are separately allocated to the RECON1, RECON2, and RECON3 file names.

The DBDLIB data sets are concatenated to the IMS file name.

Important: When you specify a generic HALDB name in the Partitioned Database panel; option 6 works only if you use four (4) or fewer DBD data sets. However, for greater flexibility you can specify up to ten (10) data sets.

Chapter 10. MSDB Maintenance utility (DBFDBMA0)

Use the MSDB Maintenance utility (DBFDBMA0) to create a z/OS sequential data set for an initial load of an MSDB database and to maintain MSDB databases.

The DBFDBMA0 utility is an offline utility that runs in a z/OS batch region.

The DBFDBMA0 utility inserts, replaces, deletes, and modifies main storage databases in the MSDBINIT file. All of these actions can be performed on different MSDB databases in a single run of the utility.

The INSERT, REPLACE, and DELETE functions of the MSDB Maintenance utility relate to entire MSDB databases. The MODIFY function is used to insert, replace, delete, and alter segments within MSDB databases. It can be used to alter one or more fields in specified segments or across a range of segments by key.

The general action of the DBFDBMA0 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 statement.
2. When the MSDB name in the old MSDBINIT file is equal to the name in the control statement, perform the actions requested in the control statement.
3. Repeat steps 1 and 2 until the input files are processed.

The figure below shows the input and output data sets used by the MSDB Maintenance utility for initializing and altering MSDBs.

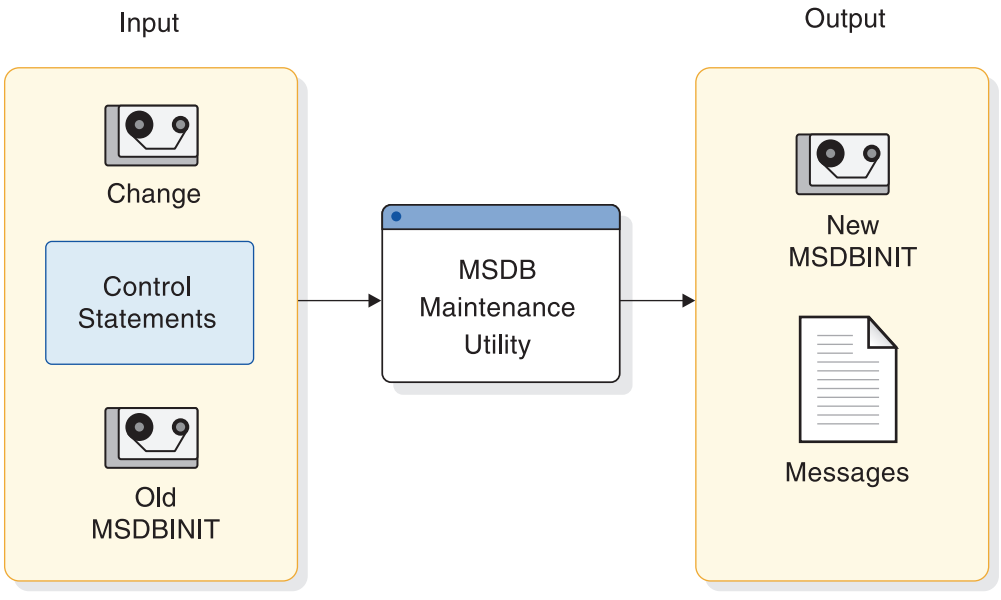


Figure 50. MSDB Maintenance utility input and output data sets

Subsections:

- "Restrictions" on page 118
- "Prerequisites" on page 118

© Copyright IBM Corp. 1974, 2015

- "Requirements"
- "Recommendations"
- "Input and output"
- "Return codes" on page 120
- "JCL specifications" on page 120

Restrictions

If you are deleting an MSDB database by specifying DELETE in the action control statement, you cannot include a change record for the MSDB database you are deleting.

If you are modify records within an MSDB database by using the DBFDBMA0 utility and an MSDB change data set, the following restrictions apply:

- A range of records cannot be inserted using the TO= keyword.
- A record (key) cannot appear more than once for the same MSDB database in the MSDB change data set.
- If you are using an MSDB change data set that uses the MSDBINIT statement format, the equal sign (=) is a reserved symbol for keywords and cannot be used in the remarks.

The DBFDBMA0 utility will work on Main Storage (MSDB) databases only.

Utilities that alter databases cannot be run while the database is quiesced.

Prerequisites

Currently, no prerequisites are documented for the DBFDBMA0 utility.

Requirements

The DBFDBMA0 utility requires at least 512 KB of virtual storage.

The DBFDBMA0 utility must be executed in a separate job step. The program cannot switch from one utility to another within the same job step.

Except for dynamic MSDB databases, which can be empty, MSDB databases must have at least one record.

Recommendations

Currently, no recommendations are documented for the DBFDBMA0 utility.

Input and output

The MSDB Maintenance utility uses various 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 table below identifies the inputs and outputs for the MSDB Maintenance utility.

Table 4. Data sets used by the MSDB Maintenance utility

Input	Output
Change	New MSDBINIT
Control	Messages
Old MSDBINIT	

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, unless you are only deleting MSDBs.

  If you are inserting a new MSDB database, it must contain data for all segments to be formatted into new MSDB databases. If you are replacing an MSDB database, you must name MSDBs that exist on the old MSDBINIT data set, and must supply data for all segments. If you are modifying an MSDB database, the change data set must supply the MSDB name, key field name, and, optionally, data for the segment to be modified.
- A control data set containing a run statement and one or more action statements specifying the function to be performed.

  If you are inserting a new MSDB database, you must specify MODE=INSERT for each MSDB to be inserted. If you are replacing an MSDB database, you must specify MODE=REPLACE for each MSDB to be replaced to appear in the new MSDBs. If you are deleting an MSDB database, you must specify MODE=DELETE for each MSDB to be deleted. If you are modifying an MSDB database, you must specify MODE=MODIFY for each MSDB to be modified.
- 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.

  If you are inserting a new MSDB, the old MSDBINIT data set, if present, must not contain a record for any MSDB to be inserted. If you are replacing an MSDB database, an old MSDBINIT data set must be supplied and must contain at least one record for each MSDB to be replaced. If you are deleting an MSDB database, you must supply an old MSDBINIT data set. If you are modifying an MSDB database, an old MSDBINIT data set must be supplied and must contain at least one record for each MSDB to be modified.

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 PROCLIB member. The MSDBABND parameter specifies that the IMS control region is to abend if an error occurs while loading an MSDB during system initialization.

You can specify the following conditions under which the IMS control region must abend during loading of the MSDBs:

- An initial checkpoint for MSDBs cannot occur.
- The MSDBs cannot be loaded due to errors with the MSDBINIT data set.
- The MSDBs cannot be loaded due to errors with the MSDBINIT data set or no segments exist in the MSDBINIT data set for at least one defined MSDB.

Return codes

The following return codes are provided by the DBFDBMA0 utility:

Code	Meaning
0	Utility executed successfully.
4	Error—error message printed.
8	Unable to open print data set.

JCL specifications

The DBFDBMA0 utility is executed as a standard z/OS job.

A Change MSDB data set can also include keywords and parameters that specify utility processing options.

The following JCL statements are required to run the DBFDBMA0 utility:
- An EXEC statement
- DD statements defining input and output
- A RUN control statement
- An ACTION control statement

EXEC statement

This statement can either specify a procedure that contains the required JCL be in the form:

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

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.

Control statements for the DBFDBMA0 utility

The MSDB Maintenance utility requires two types of control statements called the run statement and the action statement.

These are coded on a single line in columns 1 through 71. Statements cannot continue on the next line. The statements can include comments.

If you specify MODIFY in the action control statement, you must also include the MSDB change data set as input to the DBFDBMA0 utility. In the MSDB change data set, you code the keywords and parameters that specify the changes that the DBFDBMA0 utility performs on the MSDB database.

Dynamic MSDBs can be empty, but other types of MSDBs must have at least one record.

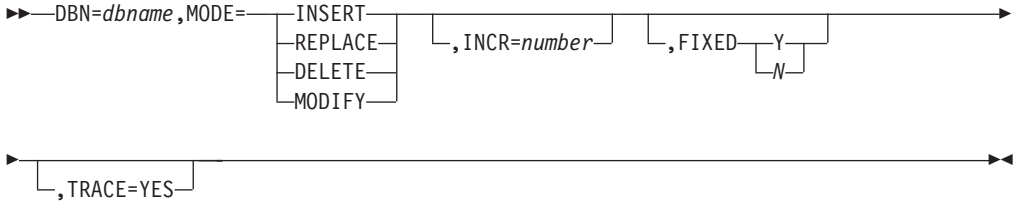
Run statement

One run statement is used for the entire utility execution, as shown.

PROC=
Specifies the 1-byte suffix for the MSDB start procedure (DBFMSDBx).

Action statement

One action statement is required for each MSDB database to be included in the utility execution. Any MSDB database 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=
The following keywords designate 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. If you specify the MODIFY keyword, you must also include an MSDB change data set.

The utility compares the MSDB 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. Inserting a range of records is not supported.

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.

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.

Restriction: A record (key) cannot appear more than once for the same MSDB database in the MSDB change data set.

MSDBINIT record format

For the MODIFY operation do the following:

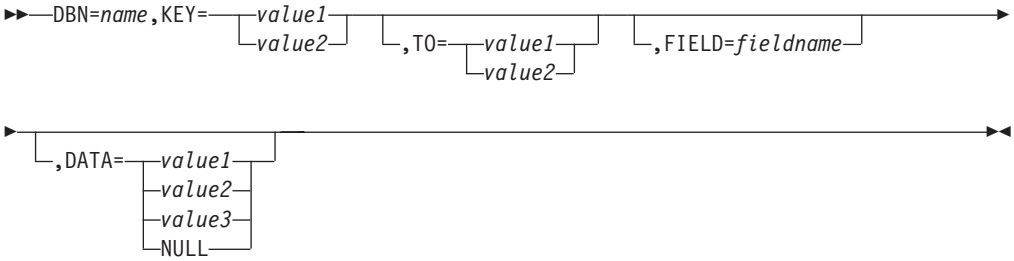
- To replace a segment, supply a record with an existent key in the KEY field and the desired segment content in the MSDB segment field.
- To insert a segment, supply a record with a nonexistent key in the KEY field and the desired segment content in the MSDB segment field.
- To delete a segment, supply a record with an existent key in the KEY field and no data in the MSDB segment field.

MSDBINIT statement 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. Comments can appear in the same statement as keywords.

Restriction: The equal sign (=) is a reserved symbol for keywords and cannot be used in comments.

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.

Examples for the DBFDBMA0 utility

These examples show the JCL necessary to execute the MSDB Maintenance utility. The control data set and the change record data set are used as input.

Subsections:

- "Two new MSDBs creation"
- "Delete, modify, and replace MSDBs" on page 125
- "Merge MSDB files" on page 126

Two new MSDBs creation

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'9FFFFFFF'     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
/*
```

Delete, modify, and replace MSDBs

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
/*
```

Merge MSDB files

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
/*
```

Related reference:

DBFMSDBx member of the IMS PROCLIB data set (System Definition)

Functions of the MSDB Maintenance utility

The following subsections describe the steps to insert, replace, delete, and modify an MSDB database by using the DBFDBMA0 utility.

Inserting MSDB databases

To insert MSDB databases:

- 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 MSDB databases

To replace MSDB databases:

1. The change data set must meet the following requirements:
   - Must be supplied
   - Must be either in card format or in MSDBINIT format or both
   - Must name MSDBs that exist on the old MSDBINIT data set
   - Must supply data for all segments
2. The control data set must contain an ACTION statement, specifying MODE=REPLACE for each MSDB to be replaced to appear in the new MSDBs.
3. 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 MSDB databases

To delete MSDB databases:

- 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 MSDB databases

To modify MSDB databases:

- 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.

Part 2. Backup utilities

Use the backup utilities to make backup copies of a database, DEDB area, or HALDB partition.

The back up copies created by these utilities are called *image copies*. Each topic introduces one utility, describes how it works, defines the requirements and restrictions for its use, and provides examples.

© Copyright IBM Corp. 1974, 2015

Chapter 11. Database Image Copy utility (DFSUDMP0)

Use the Database Image Copy utility (DFSUDMP0) to create an image copy of a database data set from a memory dump of the data set that is being copied. Use the output data set from the DFSUDMP0 utility as input to the Database Recovery utility (DFSURDB0).

You can make batch or concurrent image copies with the DFSUDMP0 utility. DBRC is required for concurrent image copies.

Multiple data sets or areas can be copied on mixed DASD devices with one execution of the DFSUDMP0 utility.

The DFSUDMP0 utility can create one or two copies of the output image copy. 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.

When creating an image copy of a shared secondary index, you need to specify the only first DBD for the DFSUDMP0 utility to copy the entire data set.

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

The following figure is a flow diagram of the Database Image Copy utility.

© Copyright IBM Corp. 1974, 2015

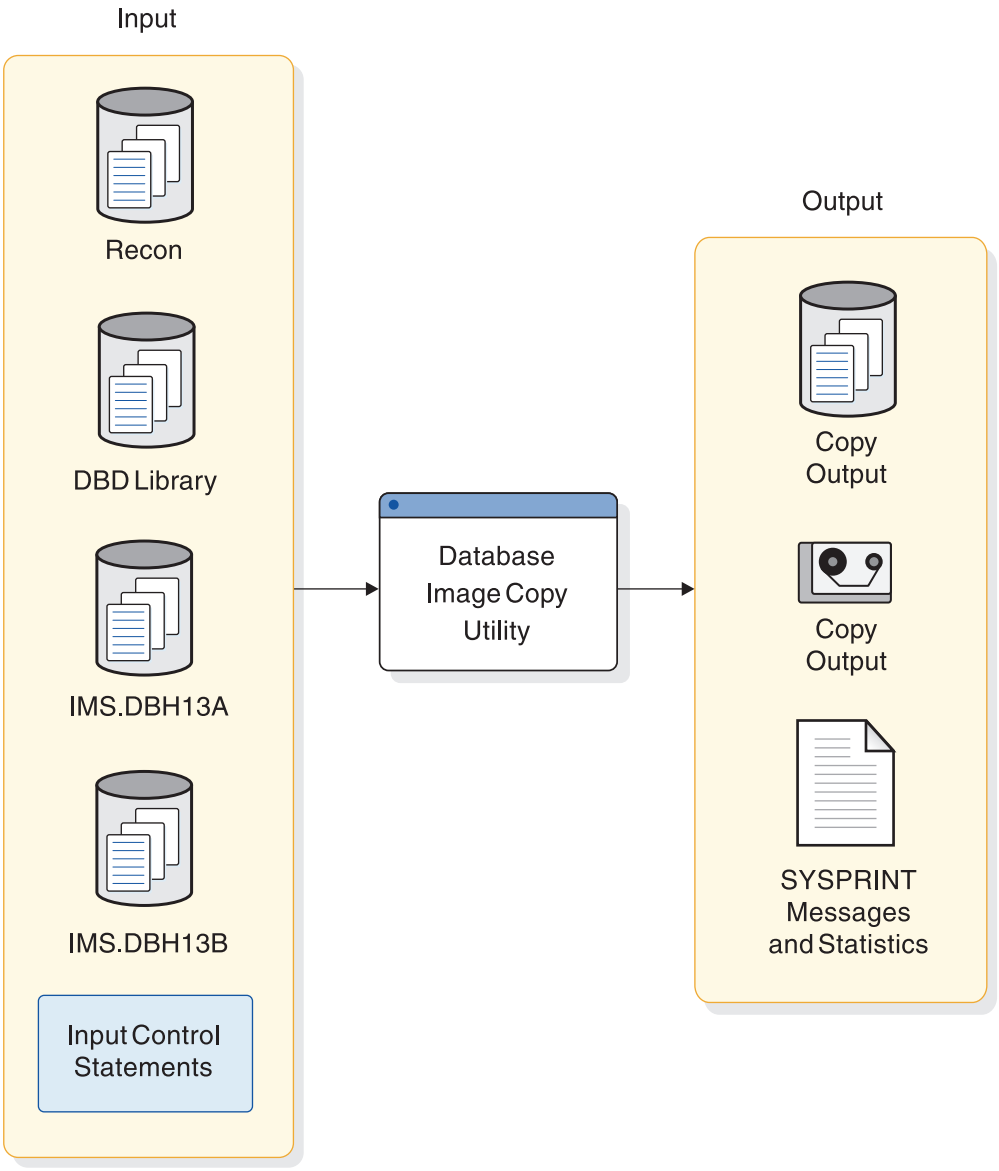


Figure 51. Database Image Copy utility

Subsections:

- "Restrictions"
- "Prerequisites" on page 133
- "Requirements" on page 133
- "Recommendations" on page 134
- "Input and output" on page 135
- "Return codes" on page 135
- "JCL specifications" on page 135

Restrictions

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

- HSAM, GSAM, and MSDB databases cannot be copied with this utility.

- If you are making a clean image copy, the database being copied cannot 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.) Alternately, a database quiesce command can be issued with the OPTION(HOLD) to put the database in a QUIESCE HELD state.
- If updates are made to the database while the Database Image Copy utility is making a copy of a VSAM KSDS, do not use the image copy for a recovery of the database.
- Concurrent image copies (fuzzy image copies):
  - Are not supported for VSAM KSDS database data sets.
  - Are not supported for databases or areas registered with share level 0.
  - Cannot be created for a database with an access level of EX (EXCLUSIVE).
  - Cannot be created for a nonrecoverable database. If a CIC of a nonrecoverable database is attempted, it fails with the message DSP0090I.
  - Cannot be created for a partition data set while the integrated HALDB Online Reorganization function is reorganizing the partition.
- The DFSUDMP0 utility cannot be restarted.
- When running an image copy utility in IMS systems that use HALDB Online Reorganization, additional restrictions apply.
- 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. Utilities that alter databases cannot be run while the database is quiesced.

Prerequisites

Depending on the circumstances in which you are using the DFSUDMP0 utility, you must meet certain prerequisites before running the DFSUDMP0 utility.

Prerequisites for the DFSUDMP0 utility include:

- If you are creating a clean image copy of a database data set or area, you must stop access to the database, area, or partition containing the data set before running the DFSUDMP0 utility. You can stop access to a database, area, or partition by issuing any of the /DBR, /DBD, or UPDATE...STOP(ACCESS) commands for the database, area, or partition. Wait for message DFS0488I before starting the Database Image Copy utility. You are not required to stop the database or partition if you are creating a concurrent image copy.
- If a write error occurs in a database, you must recover the database before you run the Database Image Copy utility. If the database is registered with DBRC and the DFSUDMP0 utility uses DBRC, then DBRC prevents the execution of the DFSUDMP0 utility.

Requirements

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

- Standard tape labels must be used on all output copies created by the Image Copy utility because logical record lengths and blocking factors are calculated at execution time.
- If you are creating an image copy of a HALDB partition data set, DBRC must be active or the copy request is rejected.

- If you are creating an image copy of a multiple DEDB area data set, the area name you specify in the control statement must be registered in the DBRC RECON data set.
- If you insert or delete one or more areas into a DEDB database by running DBDGEN while the database is stopped, you must take image copies of all areas that follow the inserted or deleted areas before the database is started again.
- 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

  Tip: Use system-determined block sizes.
- If DISP=OLD is used with the data set that is used to create a batch image copy and the database is in a QUIESCE HELD state, you must specify DISP=SHR.

Recommendations

The DFSUDMP0 utility has several recommendations related to its use and execution:

- Performance can be enhanced by providing additional buffers through the appropriate job control statements. The optimum number of buffers depends on your particular requirements. If you are copying a VSAM data set, performance improves when the AMP parameter is used to specify additional VSAM buffers. If you are copying an OSAM data set, you can specify DCB=BUFNO=*n*, where *n* is the number of blocks/CI per track.
- Use system-determined block sizes, particularly if you are copying block sizes greater than 32760 bytes on devices that support the larger block sizes.
- If you are copying multiple data sets or areas with one execution of the Image Copy utility, copy all data sets of a database at the same time. For DEDBs, if a multiple area data set (MADS) is registered in the DBRC RECON data set, you can specify the MADS as input to the Image Copy utility.
- 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 that are created after the previous execution of the batch jobs would not be valid after the reprocessing. Therefore, you cannot attempt a recovery by starting with an image copy, applying log tapes, reprocessing unlogged batch executions, then applying more log tapes.
- In Remote Site Recovery (RSR) environments, 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 that is defined during IMS system definition is used.

Input and output

The primary input to the DFSUDMP0 utility is a database data set to be copied. The primary output of the DFSUDMP0 utility is an image copy of the database data set that can be used as input to the Database Recovery utility (DFSURDB0).

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.

The following table identifies inputs and outputs for the Database Image Copy utility.

Table 5. Input to and output from the Database Image Copy utility

Input	Output
RECON	SYSPRINT messages and statistics
DBD library	Image copy data sets
One or more database data sets to be copied	
Utility control statements	

Return codes

The Database Image Copy utility provides the following return codes:

Code	Meaning
0	All operations completed successfully.
4	Warning messages were issued.
8	One or more operations were not successful.
16	Severe errors caused the job to terminate before completing all operations.

These return codes can be tested by the COND parameter on the EXEC statement of a subsequent job step.

JCL specifications

The DFSUDMP0 utility is executed as a standard z/OS job. The JCL specifications for the DFSUDMP0 utility include a JOB statement, the EXEC statement, and the DD statements. One or more utility control statements are required.

The following JCL statements are required:

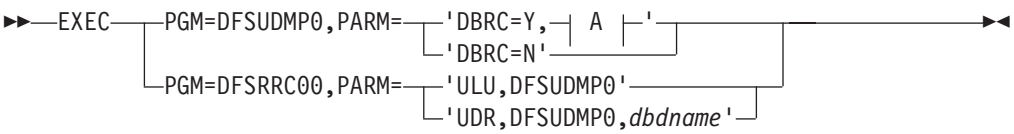
- A JOB statement that you define to meet the specifications of your installation.
- An EXEC statement
- DD statements that define inputs and outputs

One or more utility control statements must also be submitted with the JCL statements.

EXEC statement

The EXEC statement for the DFSUDMP0 utility can either invoke a cataloged procedure or can be specified in the JCL by using a specific format.

If you specify the EXEC statement in the JCL, it must use the following format:



A:

The following list describes the parameters that you can include in the EXEC statement for DFSUDMP0:

CIC
Creates a concurrent image copy of OSAM and VSAM ESDS database data sets.

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 share level must be specified in order to run CIC. You can use share level 1, 2, or 3. The DBRC share level is specified on the INIT.DB command.

The access level is set during initialization of the database during system definition by using the DATABASE macro or the CREATE DB or the UPDATE DB commands.

Restrictions:

- CIC is not supported for VSAM KSDS database data sets.
- CIC cannot be specified for databases or areas registered with share level 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.

DFSUDMP0
Specifies that the DFSUDMP0 utility is executed independently of the IMS region controller, which is the normal execution mode of the DFSUDMP0 utility. Mutually exclusive with DFSRRC00.

For example, a stand-alone execution of the DFSUDMP0 utility is specified by EXEC DFSUDMP0,DBRC=Y.

DBRC
Specifies whether the DFSUDMP0 utility uses DBRC.

PARM='DBRC=' can be specified as Y or N to override the default DBRC specification. DBRC is used during the execution of this utility if DBRC=Y is specified in either the DFSPBxxx member or the DFSIDEF0 installation defaults module, 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 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 is specified in the installation defaults module DFSIDEF0, 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.

Important: If you change DBRC parameters through a control blocks generation, but do not re-link the IMS nucleus before running the DFSUDMP0 utility, then the two forms of execution specified by the DFSUDMP0 and DFSRRC00 parameters are different.

If the DFSUDMP0 utility is executed as a stand-alone utility independently of the IMS region controller (as specified by EXEC DFSUDMP0,DBRC=Y), the new DBRC values are loaded from their control blocks module from IMS.SDFSRESL. If the DFSUDMP0 utility is executed as an IMS region (as specified by EXEC PGM=DFSRRC00,PARM='ULU,DFSUDMP0'), the old DBRC values within the IMS nucleus are used.

DFSRRC00
Specifies that the DFSUDMP0 utility is executed using the Region Controller program (DFSRRC00).

Mutually exclusive with DFSUDMP0. This parameter is supported with the subparameters ULU and UDR for upward compatibility purposes.

For example, an execution of the DFSUDMP0 under the batch region controller is specified by EXEC PGM=DFSRRC00,PARM='ULU,DFSUDMP0'

The normal execution mode for the DFSUDMP0 utility is specified by the DFSUDMP0 parameter.

GSGNAME
A optional 1 to 8-character name that identifies the global service group.

IMSPLEX
Specifies which IMSplex DBRC should join.

The IMSPLEX parameter can be specified on all job steps that use DBRC.

DBRCGRP
Specifies the DBRC group ID defined in the RECON data set used by the DBRC group.

ULU
Specifies a load/unload region.

This parameter is supported for upward compatibility and can only be specified when the DFSUDMP0 utility is executed using the Region Controller program (DFSRRC00). If this parameter is used, then the normal IMS positional parameters can be used.

UDR
Specifies a recovery region. When PARM=UDR is specified, a valid dbdname is required, but is ignored by the Image Copy utility.

This parameter is supported for upward compatibility and can only be specified when the DFSUDMP0 utility is executed using the Region Controller program (DFSRRC00). If this parameter is used, the normal IMS positional parameters can follow.

DD statements

The DFSUDMP0 utility uses a number of required and optional 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 copied by a dump. 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. When it is omitted, the utility allocates the data set dynamically with a DISP=SHR. 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. 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.

Related concepts:

Database quiesce (Database Administration)

Initializing and maintaining the RECON data sets (System Administration)

Related tasks:

Image copies in an RSR environment (Database Administration)

Related reference:

Chapter 12, “Database Image Copy 2 utility (DFSUDMT0),” on page 143

Chapter 13, “Online Database Image Copy utility (DFSUICP0),” on page 169

DBBBATCH procedure (System Definition)

DLIBATCH procedure (System Definition)

Control statement for the DFSUDMP0 utility

The DFSUDMP0 utility requires you to include a utility control statement.

The DFSUDMP0 utility control statement 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.

Examples for the DFSUDMP0 utility

These examples show uses for the DFSUDMP0 utility.

The examples in this topic 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 topic, if you are using DBRC without dynamic allocation, you must add the DD statements to the sample JCL, as shown in the figure below:

Figure 52. DD statements for using DBRC without dynamic allocation or when the database is in a quiesce held status

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

Subsections:

- "Copy and rename"
- "Copy data sets"
- "Copy an area" on page 142

Copy and rename

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
```

Copy data sets

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
```

Copy an area

In this example, the area with the area name AREANAM1 is to be copied from the database named DI32DB01. The ddnames and data set names 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 12. Database Image Copy 2 utility (DFSUDMT0)

Use the Database Image Copy 2 utility to take image copies of IMS databases by using one of the following functions of the Data Facility Storage Management Subsystem (DFSMS): the concurrent copy function or the fast replication function.

The concurrent copy function and the fast replication function of DFSMS are both hardware and software solutions that enable 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 DFSMS concurrent copy function invokes the DFSMSdss command DUMP CONCURRENT and requires hardware such as an Enterprise Storage Server® (ESS), a 3990 Storage Control, or a RAMAC Virtual Array (RVA) with Snapshot capability, or an equivalent device.

The DFSMS fast replication function invokes the DFSMSdss command COPY FASTREP(REQ) and requires hardware such as an Enterprise Storage Server (ESS) with FlashCopy® capability or a RAMAC Virtual Array (RVA) with Snapshot capability.

Subsections:

- "Restrictions"
- "Prerequisites" on page 144
- "Requirements" on page 144
- "Recommendations" on page 145
- "Input and output" on page 145
- "Return codes" on page 146
- "JCL specifications " on page 146

Restrictions

Some restrictions for the Database Image Copy 2 utility apply generally to the utility regardless of which DFSMS copy function you are using and other restrictions apply only to a specific copy function.

General restrictions

The following general restrictions apply when using the Database Image Copy 2 utility, regardless of the copy function you choose:

- HSAM, GSAM, or MSDB databases cannot be copied with this utility.
- When running an image copy utility with HALDB Online Reorganization, additional restrictions apply.
- Utilities that alter databases cannot be run while the database is quiesced.

Concurrent copy function restrictions

Currently, no restrictions are documented specific to the concurrent copy function of the Database Image Copy 2 utility.

© Copyright IBM Corp. 1974, 2015

Fast replication function restrictions

The following restrictions apply only if you are using the fast replication function of the Database Image Copy 2 utility:

- Fast replication produces only one output copy
- Fast replication does not support the same data set (SAMEDS) options of the Database Image Copy 2 utility.
- Fast replication does not support the DFSMSdss COMPRESS parameter.
- The DBRC REUSE attribute is not supported for fast replication image copy data sets.

Prerequisites

Currently, no prerequisites are documented for the DFSUDMT0 utility

Requirements

The Database Image Copy 2 utility has both general and function-specific requirements for operation.

General requirements

The following general requirements apply when you use the Database Image Copy 2 utility, regardless of the copy function you choose:

- To take advantage of the Database Image Copy 2 utility, databases must be registered with DBRC. DBRC must be active when you run the Database Image Copy 2 utility.
- Databases and area data sets that are to be copied must reside on hardware that supports the function of the Database Image Copy 2 utility that you are using.
- If multiple DBDS Select statements are submitted to the Database Image Copy 2 utility for the multiple DBDSs of either a full-function non-HALDB database or of a HALDB partition, all statements must specify the same image copy function: either fast replication or DFSMS concurrent copy.
- Nonrecoverable databases and areas must be stopped before using the Database Image Copy 2 utility. Because updates are not logged, nonrecoverable databases cannot be active during image copy processing. Utility control statements must specify X (for exclusive access).
- The DBDS 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 in an integrated catalog facility (ICF) and must satisfy one of the following requirements:

- SMS-managed.
- Cataloged using an alias (That is, the high-level qualifier of the data set name is an alias for the catalog).
- Cataloged in the master catalog.

To copy a VSAM key sequenced data set (KSDS) while it is being updated, the data set must be SMS-managed and BWO(TYPEIMS) must be specified on the AMS DEFINE or ALTER statement.

To copy a VSAM KSDSs that is not SMS-managed or for which BWO(TYPEIMS) is not specified, the KSDS database data set must be stopped before executing the utility.

When running an image copy utility with HALDB Online Reorganization, additional requirements apply.

Concurrent copy function requirements

The following requirements apply only if you are using the DFSMS concurrent copy function of the Database Image Copy 2 utility.

If DISP=OLD is used with the data set that is used to create an image copy while the database is unavailable for update processing until the image copy is physically complete (SMSNOCIC with DBREL(P)) and the database is in a QUIESCE HELD state, you must specify DISP=SHR.

The databases and area data sets that are to be copied must reside on the following kinds of hardware that supports the DFSMS Concurrent Copy feature:

- 3990 Storage Control Model 3, extended function with licensed internal code
- equivalent device

Fast replication function requirements

The following requirements apply only if you are using the DFSMS fast replication function of the Database Image Copy 2 utility:

The input DBDSs or area data sets that are to be copied must reside on hardware that supports the DFSMS Fast Replication feature (such as IBM Enterprise Storage Server hardware and its FlashCopy function or the IBM RAMAC Virtual Array hardware and its SnapShot function). The output data sets for fast replication image copies must be cataloged.

Recommendations

Currently, no recommendations are documented for the DFSUDMT0 utility

Input and output

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.

DFSMS requires that you do not specify the BUFNO keyword for either the input or output data sets.

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

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

An output image copy created by the concurrent copy function of the Database Image Copy 2 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.

An output image copy created by the fast replication function of the Database Image Copy 2 utility is an exact copy of the original data set, rather than the standard batch image copy format or the DFSMS dump format. The copy is registered with DBRC as an SMSOFFLC or SMSONLC image copy, depending on whether the image copy is a clean or fuzzy image copy.

If you are using the concurrent copy function, you can create up to four output image copies; however, only the first two are registered in the RECON data set. The advantage of specifying two or more copies is that if an I/O error occurs during copy execution, the utility continues until completion on the remaining copies.

If you are using the fast replication function, the utility can create only a single output image copy.

Return codes

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.

JCL specifications

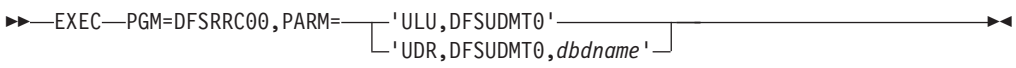
The Database Image Copy 2 utility is executed as a standard z/OS job. The following JCL statements are required:

- A JOB statement that you define to meet the specifications of your installation.
- An EXEC statement
- DD statements that define inputs and outputs

The JCL specifications for the Database Image Copy 2 utility can also include utility control statements.

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.

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.

DFSMFIMS DD
Defines the input control statement data set. This data set can reside on a tape, direct-access volume, or be routed through the input stream {DD * or DD DATA}. Use this DD statement when you need to associate the time that is spent executing DSS for an IMS IC2 job name or account identifier. The DFSMFIMS DD statement is optional.

The DFSMFIMS DD statement includes one of the following utility control statements, starting in column one: TYPE30=*jobname* or TYPE30=*'alphanumeric_string'*.

jobname
A keyword followed by a blank space. Comments are allowed after the blank space.

alphanumeric_string
A 1- to 8-byte alphanumeric string enclosed in single quotation marks. The first character is alphabetic: A-Z. The ending quotation mark is followed by a blank space. Comments are allowed after the blank space.

Within the single quotes, no leading, embedded, or trailing blanks are allowed.

The *jobname* keyword or *alphanumeric_string* value appears in the SMF type 30 accounting record.

The DFSMFIMS DD statement passes the *jobname* record or the *alphanumeric_string* value to DFSMFDSS as the server name. Parameters that are input in lowercase are translated to uppercase. The jobname keyword or *alphanumeric_string* value appears in the SMF type 30 accounting records that are produced by DFSMSDSS.

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 a HALDB data set, if this DD statement is present, the disposition specified will be honored by the utility. Otherwise, DISP=SHR is the default. If the DD statement contains a DD DUMMY, a DSN=NULLFILE, or an inactive data set, the utility dynamically allocates the active data set using the disposition specified in the DD statement.

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 output data set.

This specification can include:

- output data set name
- output data set attributes
- output data set volsers

If you are using the concurrent copy function and you are creating multiple output copies, the *dataout1* DD statement defines the first output copy.

For the fast replication function, the *dataout1* DD statement is optional for non-SMS managed data sets, if an HLQ specification statement for the output data set has been included. If the *dataout1* DD statement is included, it is used only to specify the volser information for the non-SMS managed output data set. The output data set attributes are extracted from those of the input data set, and the output data set name is generated by the HLQ specification. For SMS-managed data sets, the attributes are taken from the input data set.

The ddname can be any 1- to 8-character string, but the ddname must appear in the associated utility control statement. ESS and RVA are the only valid devices which can be used to take flash copy.

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

dataout2 **DD**
Required only if you are using the concurrent copy function and 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.

The fast replication function does not support multiple output copies of a data set.

dataout3 **DD**
Required only if you are using the concurrent copy function and 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.

The fast replication function does not support multiple output copies of a data set.

dataout4 **DD**
Required only if you are using the concurrent copy function and 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.

The fast replication function does not support multiple output copies of a data set.

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.

JCL examples

The following figure shows the concurrent copy function utility control statements 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 is issued for the group GROUPXYZ. This example is valid only if none of the output data sets resides on the same tape volume as any of the other output data sets.

Figure 53. Specifying a data group with the concurrent copy function

```
|...+....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
/*
```

The following figure shows the concurrent copy function utility control statements used to copy DBDSs for four different databases.

Figure 54. Stacking image copies in a single output data set when using the concurrent copy function

```
|...+....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
/*
```

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.

The following figure shows the JCL for the Database Image Copy 2 utility using the concurrent copy function.

Figure 55. Database Image Copy 2 utility JCL using the concurrent copy function

```
//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
```

Related concepts:

Making database backup copies (Database Administration)

Reorganizing HALDB databases (Database Administration)

Related reference:

z/OS: DFSMSdss - Concurrent copy

DBBBATCH procedure (System Definition)

DLIBATCH procedure (System Definition)

z/OS: MVS record type 30

Control statements for the DFSUDMT0 utility

The Database Image Copy 2 utility has 5 control statements. All of the DBDSs 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 within the same database must be the same. Authorization is done at the database or area level.

Each of the utility control statements has a fixed format using column positions. Syntax errors in either the Group or DBDS Select statements result in termination of the utility.

Subsections:

- "Concurrent copy function DBDS select statement"
- "Fast replication DBDS select statement" on page 153
- "Group name statement" on page 155
- "HLQ specification statement for the fast replication function" on page 157
- "SET PATCH control statement" on page 159

Concurrent copy function DBDS select statement

Each concurrent copy DBDS select statement identifies one DBDS that is to be copied. The DBDS select statement 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 sets 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 - 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.

Fast replication DBDS select statement

Each fast replication DBDS select statement identifies one DBDS that is to be copied. The statement contains the number of output copies and provides 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 Leave this position blank for fast replication.

2 Number of copies

The only valid value for fast replication is "1" or blank, because fast replication produces only one output copy.

If any value other than 1 is specified, the image copy fails and message DFS3158A is issued with reason code 4.

3 Ignored

4-11 dbdname

This must be the name of the physical DBD that includes the name of the data set to be dumped.

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

OUTPUT DD name

Specifies the DD name of the output data set identified in a corresponding DD statement.

Whether you must specify a DD name in positions 22–29 is dependent on the following:

- If your output data sets are SMS managed and you specify a high-level qualifier for the data set name in the HLQ specification statement, leave positions 22–29 blank and do not include a corresponding DD statement.
- If your output data sets are SMS managed, but you do not specify a high-level qualifier for the data set name in the HLQ specification statement, you must specify a DD name and include a corresponding DD statement that defines the output data set.
- If your output data sets are not SMS managed and you specify the high-level qualifier for the data set name in the HLQ specification statement, you must enter a DD name and include a DD statement that specifies only the target volumes to which the data set must be allocated. You do not need to specify the data set name on this DD statement.
- If your output data sets are not SMS managed and you do not specify a high-level qualifier for the data set name in the HLQ specification statement, you must specify the DD name and include a DD statement that specifies both a data set name and the target volumes

30–57 These positions are ignored on the fast replication DBDS select statement.

58 Copy integrity control for DBDS

Specifies whether databases can be updated during the image copy. An X (exclusive) indicates that the databases cannot be updated during the image copy process. The image copy is recorded in the RECON data set as SMSOFFLC. S (shared) indicates that the database can be updated during the image copy process. The image copy is recorded in the RECON as an SMSONLC 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.

An SMSOFFLC image copy can be used as input to the IMS High Performance Pointer Checker for z/OS while an SMSONLC image copy cannot.

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

59–61 These positions are ignored on the fast replication DBDS select statement.

62 Copy method

You must enter "F" in this position to select the DFSMS fast replication function.

63 HLQ Indicator

Valid values are "H" or blank.

An "H" in this field indicates that:

- An HLQ specification statement immediately follows this statement. The HLQ specification statement specifies the high-level qualifier used in a dynamically generated image copy data set name.
- The output data set is dynamically created by DFSMSdss.
- An output data set name is not supplied on a DD statement in the JCL.
- If a DD statement is used to specify volume serial information, columns 22 - 28 will contain the name of that dataout1 DD statement.

64–72 Unspecified in this 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 Database Image Copy 2 utility, a group name control statement must be supplied as the first control statement. Only one Group name statement is allowed in the input stream for a single job step.

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.

Options specified on the group name statement apply to all DBDSs being copied. The parameters specified or defaulted in positions 58-62 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. Valid values are X or S. The default is S.

An X (exclusive) indicates that the databases cannot be updated during all or part of the image copy process. For the concurrent copy function this might be for the duration of the logical copy phase or during both the logical and physical copy phases, depending on the specification in position 59. In either case, the image copy is recorded in the RECON as an SMSNOCIC image copy. For the fast replication function, an X indicates that the databases cannot be updated until the image copy process is complete. The image copy is recorded in the RECON data set as an SMSOFFLC image copy.

An S (shared) indicates that the databases can be updated during the image copy process. For the concurrent copy process, the image copies are recorded in the RECON as an SMSCIC image copy. For the fast replication function, the image copy is recorded in the RECON data set as an SMSONLC image copy.

Important: 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.

Remember: 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 For the concurrent copy function, notify and hold control for a 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
- The fast replication function is used, as specified by an F in position 62 of the group name statement.

A P specifies that the database is available for update after the physical copy completes. An 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.

The fast replication function does not support the COMPRESS option. If fast replication is specified by an “F” in position 62, this position is ignored.

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 Copy method.

F or blank. Enter F to select the fast replication function. Leave blank for the concurrent copy function.

63–72 Unspecified in this IMS release.

73–80 Available for user comments.

HLQ specification statement for the fast replication function

The HLQ specification statement defines the high-level qualifier for dynamically generated output data set names and specifies whether a time stamp is to be appended as a low-level qualifier. The HLQ specification statement is valid only for fast replication image copies.

The HLQ specification statement has the following requirements:

- An H must be specified in position 63 of the fast replication DBDS select statement to which the HLQ specification statement applies.
- The statement must immediately follow the DBDS select statement to which the HLQ specification statement applies.

When HLQ is used for fast replication, the dynamically generated IC data set could be either *ichlq.dbdname.ddname.Dyyddd.Thhmmss* or *ichlq.dbdname.ddname*. If "Y" is not specified for position 3, the format of IC data set is *ichlq.dbdname.ddname*.

The format of the data set name depends upon the additional specification of the time stamp specification for position 3. If the time stamp specification for position 3 is a value of 'N' or omitted, the format is *ichlq.dbdname.ddname*. If the time stamp specification for position 3 is a value of 'Y', the format is *ichlq.dbdname.ddname.Dyyddd.Thhmmss*

When an output data set name is dynamically generated by using an HLQ specification statement, the format of the data set name is: *ichlq.dbdname.ddname.Dyyddd.Thhmmss*, where:

- *hlq* is the high-level qualifier defined in positions 4–29 in the HLQ specification statement.
- *dbdname* is the DBD name of the database to be copied (minimum of 1 character).
- *ddname* is the DD name of the output data set group or area (minimum of 1 character).
- *Dyyddd.Thhmmss* is the optional time stamp specified in position 3 of the HLQ specification statement.

Table 6. Position and parameter descriptions for the fast replication HLQ specification statement

Position	Description
1	Specifies the statement type. A value of H is required.
2	Identifies the output data set to which the HLQ specification statement applies. The valid values are "1" or blank, because fast replication produces only one output copy. A blank defaults to 1.
3	Time stamp specification for low-level qualifier. The valid values are: **Y** A time-stamp trailer is appended to the output data set name as a low-level qualifier. For example, the resulting format of the output data set name would be: ichlq.dbdname.ddname.Dyyddd.Thhmmss **N** A time-stamp trailer is not appended to the output data set name. For example, the resulting format of the output data set name would be: ichlq.dbdname.ddname
4–29	High-level qualifier specification. Defines the character string used as the high-level qualifier in the dynamically generated output data set name.
30	Blank
31–38	Storage class for fast replication image copy data sets Optionally, specify a storage class for the allocation of the image copy data set. This specification is used as input to the Automatic Class Selection (ACS) routines. You must have proper RACF® authorization for the specified storage class. If columns 31-38 are blank, no storage class is used. This keyword is valid only for the DFSMS fast replication option of the Database Image Copy 2 utility.
39	Blank

Table 6. Position and parameter descriptions for the fast replication HLQ specification statement (continued)

Position	Description
40–47	Management class for fast replication image copy data sets Optionally, specify a management class for the allocation of the image copy data set. This specification is used as input to the ACS routines. You must have proper RACF authorization for the specified management class. If columns 40-47 are blank, no management class is used. This keyword is valid only for the DFSMS fast replication option of the Database Image Copy 2 utility.

SET PATCH control statement

The SET PATCH control statement sets DFSMSdss processing options for the current execution of the Database Image Copy 2 utility. The control statement allows you to specify DFSMSdss SET PATCH command offsets and values.

You can include up to 10 SET PATCH specifications in each SET PATCH control statement. If multiple SET PATCH specifications are entered, they must be contiguous. If a position is left blank, any subsequent SET PATCH specifications are ignored.

You can include multiple SET PATCH control statements.

SET PATCH control statements can appear anywhere in the control statement data stream after the Group statement, if one exists.

Each SET PATCH specification must be entered in the following format: *xxx=yy*, where *xxx* is the patch byte, and *yy* is the value to which the patch byte is to be set. If the SET PATCH control statement contains invalid syntax, processing is terminated and error message DFS3158A is issued with reason 3.

However, the Database Image Copy 2 utility does not check for invalid DFSMSdss patch byte values. For example, if *132=FF* is specified, the Database Image Copy 2 utility does not validate that 132 is a valid patch byte.

Table 7. Position and parameter descriptions for the Database Image Copy 2 utility SET PATCH control statement

Position	Description
1	Control statement identifier. “P” is the required value.
2	Blank
3–8	The first or only SET PATCH specification. This specification is required.
9	Blank
10–15	The second SET PATCH specification. This specification is optional.
16	Blank
17–22	The third SET PATCH specification. This specification is optional.
23	Blank
24–29	The fourth SET PATCH specification. This specification is optional.
30	Blank

Table 7. Position and parameter descriptions for the Database Image Copy 2 utility SET PATCH control statement (continued)

Position	Description
31–36	The fifth SET PATCH specification. This specification is optional.
37	Blank
38–43	The sixth SET PATCH specification. This specification is optional.
44	Blank
45–50	The seventh SET PATCH specification. This specification is optional.
51	Blank
52–57	The eighth SET PATCH specification. This specification is optional.
58	Blank
59–64	The ninth SET PATCH specification. This specification is optional.
65	Blank
66–71	The tenth SET PATCH specification. This specification is optional.

Related reference:

z/OS: DFSMSdss patch area

Running the DFSUDMT0 utility

Depending on the options you select, the DFSUDMT0 utility has a number of operational considerations when running the utility.

Subsections:

- "Concurrent copy function"
- "Fast replication function" on page 161
- "Multiple database data set input" on page 161
- "Specifying group names" on page 162
- "Single output data set for multiple image copies" on page 163
- "Dynamic data set names for fast replication image copies" on page 164
- "Image copy completion notification" on page 164
- "Specifying DFSMSdss SET PATCH commands" on page 166

Concurrent copy function

The concurrent copy function can make both fuzzy and clean image copies.

When using the DFSMS concurrent copy option of the Database Image Copy 2 utility, 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. After the concurrent copy session is established, the copy is 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, 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. The length of time required to complete the physical copy depends upon the size of the data set.

The physical copy is used as input by the Database Recovery utility (DFSURDB0). The physical copy time is used by DBRC to generate JCL for a recovery. If the image copy is a clean image copy (SMSNOCIC), DBRC includes the required change accumulation data sets and logs to apply changes made to the database after the stop time of the image copy. If the image copy is a concurrent image copy (SMSCIC), DBRC includes the required change accumulation data sets and logs to apply changes made to the database after the start time of the image copy.

By using the DFSMS concurrent copy function the Database Image Copy 2 utility increases database availability. You 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.

You can choose the concurrent copy option of the Database Image Copy 2 utility by specifying the concurrent copy DBDS select statement.

Fast replication function

The fast replication function uses the z/OS DFSMSdss command COPY FASTREP(REQ) and the fast replication copy technology provided by an Enterprise Storage Server (ESS) with FlashCopy capability or a RAMAC Virtual Array (RVA) with Snapshot capability.

The fast replication function uses the fast replication technology to create the image copy, and to restore the database data sets (DBDS) during recovery as well. A fast replication image copy is an exact copy of the DBDS, as opposed to the dump formatted image copy of the concurrent copy option.

Unlike the concurrent copy process, the fast replication copy process is not split into a logical copy phase and a physical copy phase. If you are creating a clean image copy, the database is unavailable for the duration of the copy process; however, the fast replication process is comparable in speed to the logical copy phase of the concurrent copy process.

When the image copy is complete, the Database Image Copy 2 utility registers it with DBRC as either a SMSOFFLC or SMSONLC image copy. If you are creating a clean image copy, the Database Image Copy 2 notifies the system console that the image copy is complete. If you are creating a fuzzy image copy, no notification is sent.

You can choose the fast replication option of the Database Image Copy 2 utility by specifying fast replication DBDS select statement.

Multiple database data set input

The Database Image Copy 2 utility can copy multiple DBDSs in one execution of the utility. You can specify multiple control statements in a single execution, one per DBDS to be copied.

The Database Image Copy 2 utility passes all of the DBDSs specified on the control statements to the DFSMSdss command on a single invocation. If you are using the concurrent copy option, DFSMSdss starts the multiple dump processes in parallel and the logical copy phase completes in a very brief period for all of the DBDSs. If you are using the fast replication option to copy multiple DBDSs, DFSMSdss starts the multiple copy processes in parallel and the entire image copy process completes more quickly than if the multiple copy requests were processed one at a 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 or, if you are using the concurrent copy function, you can use the same data set option, which concatenates multiple input data sets in a single output data set. The fast replication function does not support the same data set option.

The number of DFSMSdss dump or copy 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.

You can specify a group name to identify the set of DBDSs that are copied as a group. If you do not specify a group name, you can specify different processing options for each of the DBDSs that are copied in the same utility execution. This allows the image copies of some DBDSs copied in a single execution of the utility to be clean, or consistent, image copies while the image copies of other DBDSs copied in the same execution can be fuzzy image copies.

If you are using the concurrent copy option and you are creating a fuzzy copy of a KSDS in the same utility execution in which other DBDSs are being copied, 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.

If you are using the fast replication option, you can create fuzzy image copies of KSDSs in the same utility execution in which other DBDSs are being created without a problem. Data set types do not matter when you are using the fast replication option.

Specifying group names

You can specify a group name to represent a collection of DBDSs that are to be copied in a single execution of the Database Image Copy 2 utility by including a Group Name utility control statement.

The group names capability allows you to monitor clean image copy processing on a group basis. If you specify a group name, status is reported in the completion messages of the logical and the physical phases of the concurrent copy function, and the completion of the entire image copy process for the fast replication function.

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 or areas when the appropriate 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 copy multiple input data sets into one output data set. The concurrent copy function of the Database Image Copy 2 utility exploits this capability by providing a *same data set* option.

The same data set option concatenates the image copies of up to 255 DBDSs into a single output data set in a single execution of the utility by writing multiple dumps one after another in the output data set, each preceded by DFSMSdss dump header records. 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 single output data set containing the image copy output for up to 255 DBDS instances can be contrasted with the stacking achieved through JCL specifications, which produces multiple image copy instances in separate data sets on the same tape volume.

Concatenating the image copies by using the same data set option 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.

Restrictions: The same data set option is not supported by:

- The fast replication function of the Database Image Copy 2 utility
- DBDSs registered in the RECON data set with the REUSE attribute

To implement the same data set option, follow these steps:

1. In the first DBDS select statement that you include, define the output data set as you would normally.
2. In all subsequent DBDS select statements:

- Specify an S in column 1.
- Do not define an output data set.

Dynamic data set names for fast replication image copies

If you are using the fast replication function, you can have the Database Image Copy 2 utility dynamically generate the data set names and ddnames for your output data sets. You need only make the appropriate specification on the fast replication DBDS select statement and include an HLQ specification statement that defines a high-level qualifier. Using the HLQ specification statement, you can also have the Database Image Copy 2 utility append a time stamp to the dynamically generated names as a low level qualifier.

If you are using SMS managed DASD, DFSMSdss can create the data set itself with all of the correct allocation attributes. You do not need to include a DD statement or specify an OUTPUT ddname in the DBDS control statement.

If you are using non-SMS managed DASD, you must provide a DD statement to specify the target volumes and you must specify the OUTPUT ddname in the DBDS control statement; however, DFSMSdss can determine the rest of the allocation attributes from the source database data set.

If you choose to define your data set names and ddnames manually, you must include the output data set name in the JCL and the appropriate OUTPUT ddname in the DBDS control statement.

Image copy completion notification

When clean image copies are taken, the Database Image Copy 2 utility notifies the system console when a logical or physical phase of the concurrent copy function is complete and when the entire image copy process of the fast replication function is complete. This notification indicates that the database or group can be started.

The Database Image Copy 2 utility does not provide completion notification 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 for the concurrent copy function

The Database Image Copy 2 utility issues the DFS3121I message when an image copy created by using the concurrent copy function 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.

If the logical copy process fails for any DBDSs, DFS3121A message also identifies the failed DBDSs. When DFS3121A indicates that the image copy for a DBDS has failed, you can consider trying to copy the DBDS again before starting the group or database. For each DBDS that failed during the logical copy phase, message DFS3122A is also issued.

Physical completion notification for the concurrent copy function

If you are using the concurrent copy function to create a clean image copy and you disallow database updates until the physical copy phase is complete, the Database Image Copy 2 utility issues message DFS3141A to the system console when the image copy is physically complete. This message indicates that the image copies have been recorded in the RECON and the databases or areas can 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.

If the physical copy process fails for any DBDSs, DFS3141A message also identifies the failed DBDSs. When DFS3141A indicates that the image copy for a DBDS has failed, you can consider trying to copy the DBDS again before starting the group or database. For each DBDS that failed during the physical copy phase, message DFS3144A is also issued.

Image copy complete notification for the fast replication function

The Database Image Copy 2 utility issues message DFS3141A when a clean image copy taken using the fast replication function is complete. This message indicates that the image copies have been recorded in the RECON and the databases or areas can now be started. DFS3141A reports on the group name if a group name was specified. Otherwise, the Database Image Copy 2 utility issues a DFS3141A message for each database, HALDB partition, and area that was processed. DFS3141 messages identify the individual data sets that were successfully processed.

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 the fast replication image copy is complete for all DBDS occurrences selected for the group, or for the database, the HALDB partition, or the area.

If the fast replication copy process fails for any DBDSs, DFS3141A message also identifies the failed DBDSs. When DFS3141A indicates that the image copy for a DBDS has failed, you can consider trying to copy the DBDS again before starting the group or database. For each DBDS that failed during the physical copy phase, message DFS3144A is also issued.

Specifying DFSMSdss SET PATCH commands

This topic contains Product-sensitive Programming Interface information.

DFSMSdss provides a patch area that allows you to customize DFSMSdss processing by setting the values of patch bytes at specific offsets. The Database Image Copy 2 utility provides two methods for temporarily setting DFSMSdss patch bytes.

SET PATCH control statement

You can specify DFSMSdss processing options for the current execution of the Database Image Copy 2 utility by using the SET PATCH control statement. The control statement allows you to specify DFSMSdss SET PATCH command offsets and values to the Database Image Copy 2 utility.

You can include up to 10 SET PATCH specifications in each SET PATCH control statement. You can include multiple SET PATCH control statements.

SET PATCH commands in module DFSUDMT2

The Database Image Copy 2 utility defines generic SET PATCH commands (SET PATCH 00=00) that you can zap in module DFSUDMT2 and pass to DFSMSdss along with the DFSMSdss commands DUMP or COPY. The patches are then in effect only for the Image Copy 2 job step and not for any other DFSMSdss processing.

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.

If usage is restricted by the installation, you must ensure that authorization to issue the DFSMSdss SET command with the PATCH keyword is provided.

Related reference:

z/OS: DFSMSdss patch area

Chapter 13. Online Database Image Copy utility (DFSUICP0)

Use the Online Database Image Copy utility (DFSUICP0) 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 DFSUICP0 utility prints the time stamp of the system log when the utility starts and again when it completes.

The first record on the output image 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 (DFSURDB0) to verify input.

The DFSUICP0 utility can create one or two copies of the output image copy. 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.

The DFSUICP0 utility can take special utility checkpoints during the execution of the utility that allow the utility to restart an image copy job from the last utility checkpoint after a system or utility failure. The utility checkpoint feature is enabled by including the DFSUCKPT and DFSURSRT DD statements and coding the DD statements that define the output image copy data sets with DISP=KEEP or CATLG and nonspecific serial numbers.

This utility runs as a batch message processing program (BMP).

Subsections:

- "Restrictions"
- "Prerequisites" on page 170
- "Requirements" on page 170
- "Recommendations" on page 170
- "Input and output" on page 171
- "Return codes" on page 171
- "JCL specifications" on page 171

Restrictions

The following restrictions apply when running the DFSUICP0 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.
- In a data sharing environment, if more than one IMS subsystem has the database open for update access you cannot use the DFSUICP0 utility. As an alternative, you can use the concurrent copy option (CIC) of the Database Image Copy utility (DFSUDMP0) instead of the DFSUICP0 utility.
- The utility checkpoint restart feature cannot restart an image copy job if the job failed due to either a power failure or to an out of space condition on DASD

© Copyright IBM Corp. 1974, 2015

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.

- You cannot run the image copy utility concurrently with HALDB Online Reorganization.
- The Online Image Copy utility cannot be restarted if it fails 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.
- Utilities that alter databases cannot be run while the database is quiesced.

Prerequisites

If you are making an image copy of a nonrecoverable database, before you run the DFSUICP0 utility, 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.

Requirements

The following requirements apply when running the Online Database Image Copy utility:

- 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.

  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.
- Standard labels must be used on all output copies created by the Online Database Image Copy utility, because default block sizes are calculated at execution time if you do not specify them in the JCL.
- If you create an image copy of a database by using the DFSUICP0 utility while an application program is updating the same database, you must apply the changes made by the program when the database is recovered. Use the Database Recovery utility (DFSURDB0) to apply the changes from the logs. The time stamp of the first log data set required for recovery is printed on SYSOUT.

Recommendations

The following recommendations apply when using the DFSUICP0 utility:

- Take an image copy immediately after running batch jobs that update the database without logging. An image copy 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 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 identical, log tapes created after the previous execution of the batch jobs are not 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, and then applying more log tapes.

Input and output

The primary input to the DFSUICP0 utility is a database data set to be copied. The primary output of the DFSUICP0 utility is an image copy of the database data set that can be used as input to the Database Recovery utility (DFSURDB0).

The information in the following table identifies inputs and outputs for the DFSUICP0 utility.

Table 8. Input to and output from the DFSUICP0 utility

Input	Output
RECON	SYSPRINT messages and statistics
DBD library	Image copy data sets
One or more database data sets to be copied	
Utility control statements	

Return codes

The DFSUICP0 utility issues 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 utility checkpoints are taken.

An error received on the restart data set during restart causes a message to be printed and the utility to abnormally terminate.

JCL specifications

The JCL specifications for the DFSUICP0 utility include a JOB statement, the EXEC statement, and the DD statements. The DFSUICP0 utility is executed as a standard z/OS job. One or more utility control statements must be included with the JCL statements.

The following JCL statements are required:

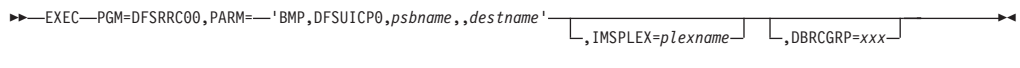
- A JOB statement that you define to meet the specifications of your installation.

- 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:



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).

IMSPLEX
Specifies which IMSplex DBRC should join.

DBRCGRP
Specifies the three-character DBRC group ID, left-justified and padded with blanks, if necessary. The DBRC group ID is passed to the DBRC SCI registration exit routine, DSPSCIX0 if it exists. The sample version of DSPSCIX0 that ships with IMS returns the value that you supply to DBRC as the DBRC group ID. This parameter is optional and can be overridden by the DBRCGRP execution parameter.

Recommendation: Use the DBRC SCI registration exit, DSPSCIX0, to determine the DBRC group ID instead of the DBRCGRP parameter.

The normal IMS positional parameters can follow.

DD statements

The DFSUICP0 utility uses a number of required and optional 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 sub-parameter 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 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.

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 utility checkpoint information specific to the execution of the utility. 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 utility 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.

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.

Related concepts:

Making database backup copies (Database Administration)

Related reference:

IMSBATCH procedure (System Definition)

Sequential buffering control statements (System Definition)

Program Specification Block (PSB) Generation utility (System Utilities)

Control statement for the DFSUICP0 utility

The DFSUICP0 utility requires you to include a utility control statement.

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. **Restriction:** The output data set cannot have the same ddname as the database being copied.
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 utility 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.

Examples for the DFSUICP0 utility

These are examples of uses for the DFSUICP0 utility.

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 to the sample JCL, as shown in the following table:

Figure 56. DD statements for using DBRC without dynamic allocation

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

This example shows the JCL and utility control statements used to copy four OSAM data set groups associated with a HIDAM database, with utility 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
/*
```

Restarting an image copy job after a DFSUICP0 failure

Information about how to restart an image copy job after a DFSUICP0 failure.

Two optional DD statements enable you to 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 utility checkpoint or restart functions are available.

The DFSUCKPT DD statement defines the data set to which the utility writes utility checkpoint information specific to the execution of the utility. The utility checkpoint information includes a volume serial number and relative record numbers. By default, if the DFSUCKPT DD statement is included, the Online Database Image Copy utility writes a utility checkpoint for every 5000 records copied. You can override the utility checkpoint interval by specifying a different interval on the control statement.

The DFSURSRT DD statement defines a utility 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 copy 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.

Related reference:

"Control statement for the DFSUICP0 utility" on page 174

Part 3. Recovery utilities

Use the recovery utilities to recover databases, Fast Path DEDB areas, and HALDB partitions.

Each topic introduces one utility, describes how it works, defines the requirements and restrictions for its use, and provides examples.

© Copyright IBM Corp. 1974, 2015

Chapter 14. Batch Backout utility (DFSBBO00)

Use the Batch Backout utility (DFSBBO00) to recover databases to a point before a program was initiated or to a checkpoint or sync point.

The DFSBBO00 utility operates as a normal IMS batch job using the PSB of the program whose errors are to be backed out and the logs of the updates made by the program.

The usefulness of the DFSBBO00 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.

The DFSBBO00 utility uses information from DBRC to validate the input log. The DFSBBO00 utility avoids performing backouts that have completed or that are done using an in-progress restart.

The DFSBBO00 utility can be run on databases at the active site in Remote Site Recovery (RSR) configurations. The DFSBBO00 utility cannot be used at the tracking site.

The following figure is a flow diagram for the DFSBBO00 utility.

© Copyright IBM Corp. 1974, 2015

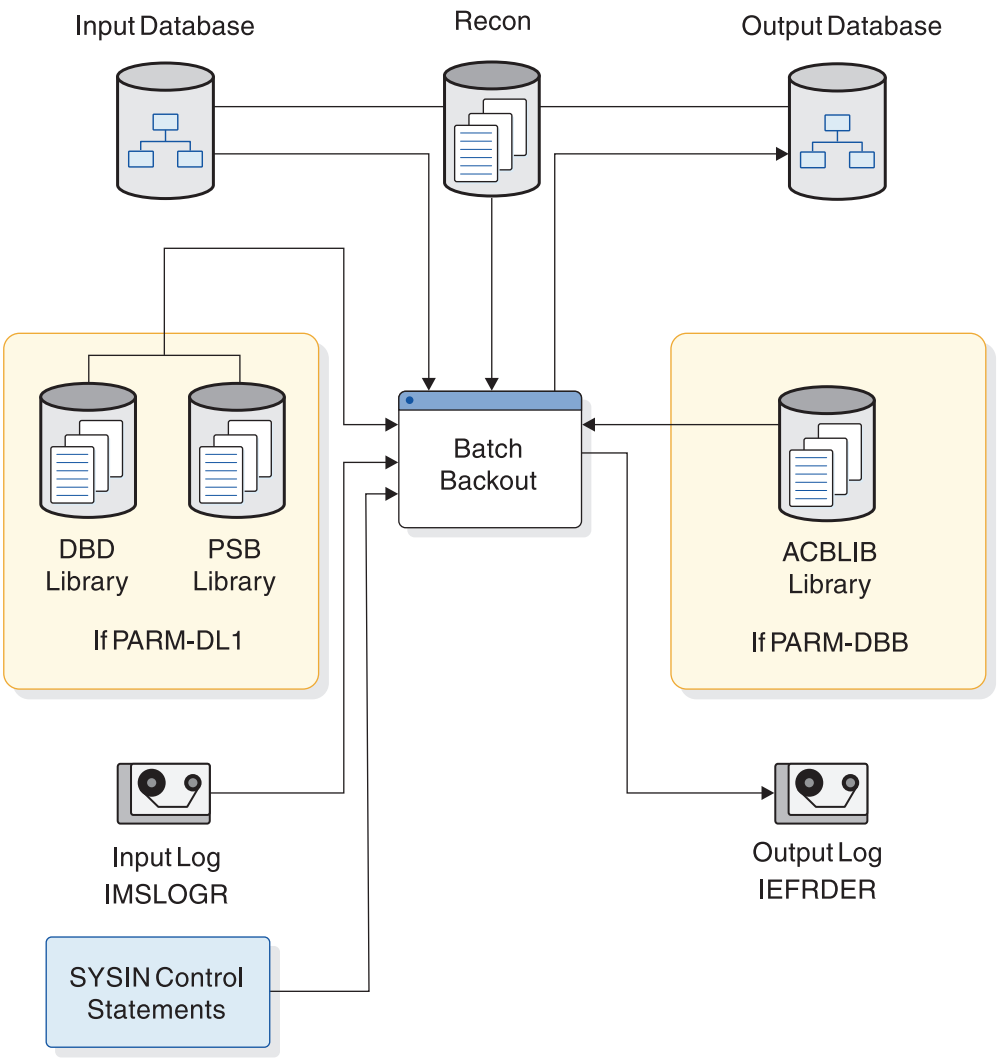


Figure 57. Data set requirements for the DFSBBO00 utility

Subsections:

- "Restrictions"
- "Prerequisites" on page 181
- "Requirements" on page 181
- "Recommendations" on page 181
- "Input and output" on page 182
- "Return codes" on page 183
- "JCL specifications" on page 185

Restrictions

The following restrictions apply when using this utility:

- The DFSBBO00 utility 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. The DFSBBO00

utility 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.

- A normally terminated job that used IRLM cannot be backed out.
- The logs used must not have been created before a reorganization.
- 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).
- The DFSBBO00 utility cannot be used at an RSR tracking site.
- The DFSBBO00 utility will work on full-function (HALDB and non-HALDB) databases only.
- Utilities that alter databases cannot be run while the database is quiesced.

Prerequisites

Before running the DFSBBO00 utility you must make sure that certain prerequisites are met.

The following restrictions apply when using this utility:

- If a write error occurs in a database that is not registered with DBRC, you must recover the database before you run the DFSBBO00 utility. If you attempt to backout without recovering first, the results are unpredictable. Without DBRC, you cannot verify that recovery completed successfully.

  When databases are registered with DBRC, you are only required to perform a recovery prior to a backout if the backout fails to complete due to an I/O error. If a backout completes normally, you can postpone recovery to a more convenient time.

Requirements

To run the DFSBBO00 utility you must satisfy various requirements.

The following requirements apply when using this utility:

- The release level of the DFSBBO00 utility must match the release level of the IMS Batch Region controller, DFSRRC00, that created the input log data sets.
- If the IMS region that created the log being used as input, used DBRC and IRLM, the DFSBBO00 utility must also use DBRC and IRLM.
- When you run the DFSBBO00 utility 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 the DFSBBO00 utility using a DLI region type to backout online reorganization PSBs.
-

Recommendations

The DFSBBO00 utility has recommendations related to its use and execution.

If you are using logs from a DB/DC or a DBCTL system and either your virtual storage area is limited or you receive an error indicating that not enough virtual private area storage is available, use the READBACK statement to tell the utility to perform the backout by reading the log backwards, instead of saving the database changes in virtual storage during the forward reading of the log.

The READBACK statement is not needed for IMS batch input logs, however, since these jobs use READBACK automatically.

Input and output

The primary input to the DFSBBO00 utility is a database that needs program updates backed out The primary output of the DFSBBO00 utility is the same database with the program updates removed. The DFSBBO00 utility can also produce an output log.

The information in the following table identifies inputs and outputs for the DFSBBO00 utility.

Table 9. Data set requirements for the DFSBBO00 utility

Input	Output
RECONs	
Databases with updates that need to be backed out	The databases that have had updates backed out
If parm=DLI, DBD/PSB libraries	IEFRDER (output log)
If parm=DBB, ACBLIB library	
One or more input logs	
SYSIN control statements	

The input to the DFSBBO00 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 the DFSBBO00 utility 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, the DFSBBO00 utility 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, the DFSBBO00 utility 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 the DFSBBO00 utility 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 that determine what is to be backed out.

The output from the DFSBBO00 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 DFSBBO00 utility might be needed as input for the Change Accumulation utility (if the time of the DFSBBO00 utility run falls within the period covered by the Change Accumulation).

Do not use this log as input to subsequent runs of the DFSBBO00 utility.

Return codes

The DFSBBO00 utility can issue any of a number of return codes. Each return code causes a message to be printed.

The DFSBBO00 utility return codes and their associated messages are:

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 (DFS3278A).

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 The DFSBBO00 utility was given a control statement which implies that it is to perform the same backout that an in-progress restart performs (DFS3292I).

124 The DFSBBO00 utility 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 and cause an ABENDU007I to be issued:

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).

JCL specifications

The JCL specifications for the DFSBBO00 utility include a JOB statement, the EXEC statement, and the DD statements. The DFSBBO00 utility is executed as a standard z/OS job. One or more utility control statements can be included with the JCL statements.

The following JCL statements are required:

- A JOB statement that you define to meet the specifications of your installation.
- An EXEC statement
- DD statements defining inputs and outputs

EXEC statement

The EXEC statement for the DFSBBO00 utility can either invoke a cataloged procedure containing the required statements or be specified in the JCL by using a specific format.

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'
```

DBB
Specifies that the DFSBBO00 utility runs in a DBB-type region. Prebuilt blocks are used. The IMSACB DD statement is required.

DLI
Specifies that the DFSBBO00 utility runs in a DLI-type region. Prebuilt blocks are not used. The IMS DD statement is required

psbname
Is the name of the PSB used by the program to be backed out.

If you are backing out a database reorganization performed by the HALDB Online Reorganization function, you must specify the eight-character PSB name used by the HALDB Online Reorganization function. PSB names for the HALDB Online Reorganization function begin with a zero followed by the seven character partition name. For example, 0POHIDKA.

nnn
Is the number of 1 KB blocks required for the database buffer pool.

The value 'C' must be specified for the DBRC parameter on the EXEC statement when backing out a normally terminated batch job that used DBRC but did not use IRLM.

If the DBRC parameter is used for a batch job whose DBRC subsystem record is marked abnormally terminated, the DFSBBO00 utility functions as if 'Y' had been specified for the DBRC parameter.

DD statements

The DFSBBO00 utility uses a number of required and optional DD statements.

Lowercase ddnames on DD statements are defined 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. The IMS DD statement concatenates the IMS.PSBLIB and IMS.DBDLIB libraries.

Attention: Generalized Sequential Access Method (GSAM) databases are not supported. The Batch Backout utility (DFSBBO00) must follow the same rules as the IMS batch job. If the IMS batch job requires the IMS DD statement because of a GSAM database in the PSB, the Batch Backout utility (DFSBBO00) requires the IMS DD statement as well, despite the fact that Batch Backout utility (DFSBBO00) does not access the GSAM database.

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 the DFSBBO00 utility 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.

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.

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 concepts:

IMS buffer pools (System Definition)

Database backout (Database Administration)

Using IMS utilities with HALDB Online Reorganization (Database Administration)

Related reference:

Sequential buffering control statements (System Definition)

DBBBATCH procedure (System Definition)

DLIBATCH procedure (System Definition)

Utility control statements for the DFSBBO00 utility

You use control statements to define the DFSBBO00 utility processing options.

Subsections

- "ABEND statement"
- "ABENDMSG statement" on page 189
- "ACTIVE statement" on page 189
- "BYPASS LOGVER statement" on page 189
- "BYPASS SEQVER statement" on page 190
- "CHKPT statement" on page 190
- "COLDSTART statement" on page 191
- "NOREADBACK statement" on page 191
- "READBACK statement" on page 192

You can provide control statements to the utility by including them in the SYSIN data set pointed to by the SYSIN DD statement.

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 DFSBBO00 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 the DFSBBO00 utility run with either the COLDSTART or the ACTIVE control statement, the DFSBBO00 utility 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 DFSBBO00 utility run in which the input log comes from an IMS DB/DC or DBCTL subsystem and no control statements are used, the DFSBBO00 utility backs out only dynamic backout failures.

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 DFSBBO00 utility 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, the DFSBBO00 utility 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 the DFSBBO00 utility 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, the DFSBBO00 utility does not compile the list. The DFSBBO00 utility 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 to. 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 DFSBBO00 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, the DFSBBO00 utility 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:

- The DFSBBO00 utility is executed without DBRC.
- The BYPASS LOGVER statement is used for the DFSBBO00 utility 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 DFSBBO00 utility when one of the following conditions is true:

- The DFSBBO00 utility is executed without DBRC.
- The BYPASS LOGVER statement is used for the DFSBBO00 utility run.
- You have not specified the ACTIVE or COLDSTART statement in a run of the DFSBBO00 utility, 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 the DFSBBO00 utility 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 - 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.

Example for the DFSBBO00 utility

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
```

Running the DFSBBO00 utility in an RSR environment

The DFSBBO00 utility can be run on databases at active sites in a Remote Site Recovery (RSR) configuration. The DFSBBO00 utility cannot be used at the tracking site.

The DFSBBO00 utility uses the Transport Manager Subsystem (TMS) to send active site log data to the tracking site, just like other IMS batch jobs. When the DFSBBO00 utility 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.

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 DFSBBO00 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 DFSBBO00 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.

Chapter 15. Database Change Accumulation utility (DFSUCUM0)

Use the Database Change Accumulation utility to streamline the recovery information you provide to the Database Recovery utility.

The DFSUCUM0 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 also sorts as a minor field RBA or key.

The DFSUCUM0 utility can be run several times over a time to incorporate additional database changes and to delete changes that are no longer useful.

The DFSUCUM0 utility can be run independently of IMS. When the DFSUCUM0 utility is run with z/OS, the output is compressed. This compression is achieved using z/OS services.

Utilities that alter databases cannot be run while the database is quiesced.

The following figure shows a flow diagram of the sources of input to the Database Change Accumulation utility and the output created by this utility.

© Copyright IBM Corp. 1974, 2015

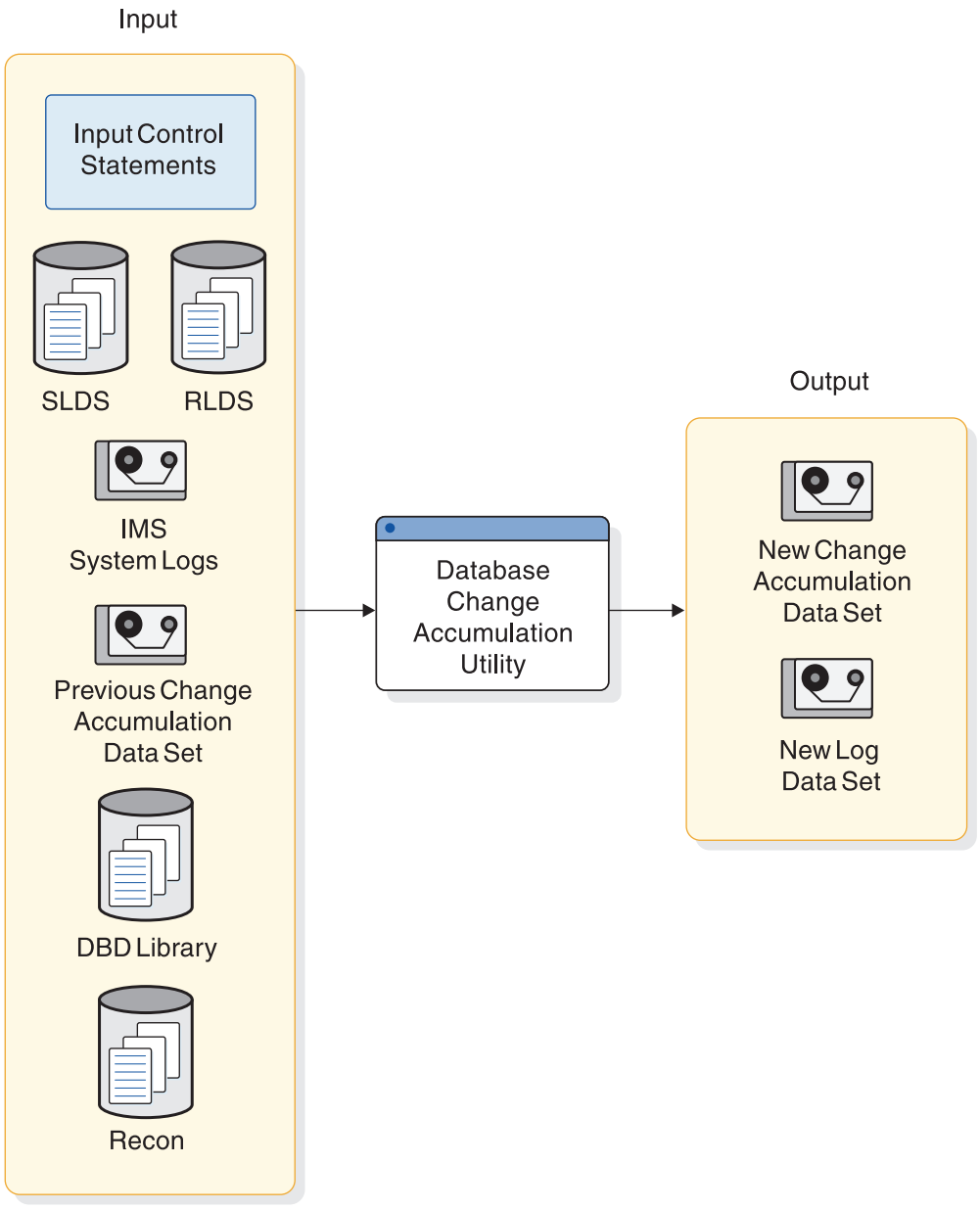


Figure 58. Database Change Accumulation utility

Subsections:

- "Restrictions"
- "Prerequisites" on page 197
- "Requirements" on page 197
- "Recommendations" on page 197
- "Input and output" on page 197
- "JCL specifications" on page 198
- "Return codes" on page 201

Restrictions

The restrictions for the DFSUCUM0 utility include:

- 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.

Prerequisites

Before running the DFSUCUM0 utility, you must archive any online log data sets (OLDS) that contain database change records that are required for recovery. You can archive the OLDS by issuing the DBRC command GENJCL.ARCHIVE to generate the JCL for the Log Archive utility (DFSUARC0) and then running the DFSUARC0 utility.

The DFSUCUM0 utility starts the z/OS SORT/MERGE program, which is an execution prerequisite.

Requirements

Log input to the DFSUCUM0 utility must be in the sequence in which it was created.

Recommendations

Use the DBRC command GENJCL.CA to generate the JCL for the Change Accumulation utility. Using the GENJCL.CA command ensures that the correct UTC purge time is applied and that the utility control statements are coded correctly for all change accumulation processing.

Input and output

The primary input to the DFSUCUM0 utility is log data sets. The primary output of the DFSUCUM0 utility is a data set containing accumulated database change records that have been optimized for recovery. The DFSUCUM0 utility also produces additional output.

The input to the Database Change Accumulation utility consists of:

- All SLDS or RLDS created since either the last image copy utility execution or the last run of this utility. This input can include the new log output data sets resulting from previous executions of this utility. New log output data sets are described in the output section.
- The previous database Change Accumulation utility data set. This data set 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.

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 reformatted database change log records identified by the database or data sets on a DB1 control statement that specifies *OTHER. New log data sets are used in a subsequent CA execution. New log data sets are not recorded in RECON. Subsequent change accumulation data sets

produced using a new log data set must be recorded in the RECON data set by a DBRC NOTIFY.CA command. The use of new log data sets is not recommended, but they are typically created and used in systems without DBRC support.

- If you are recovering a VSAM data set, message IEC161I, which is a normal message during an IMS database recovery of a VSAM data set.

The information in the following table depicts the sources of input to the Database Change Accumulation utility and the output created by this utility.

Table 10. 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	

JCL specifications

The DFSUCUM0 utility is run as a standard z/OS job. The JCL specifications for the DFSUCUM0 utility include a JOB statement, an EXEC statement, and DD statements. One or more utility control statements are required.

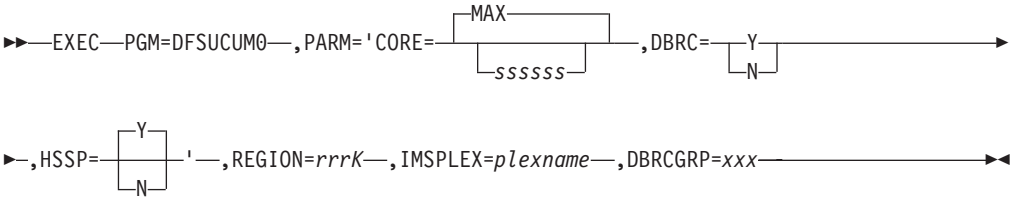
The following JCL statements are required:

- A JOB statement that you define to meet the specifications of your installation.
- An EXEC statement
- DD statements that define inputs and outputs

EXEC statement

The EXEC statement for the DFSUCUM0 utility can either start a cataloged procedure containing the required statements or be specified in the JCL by using a specific format.

If you specify the EXEC statement in the JCL, it must be in one of the following forms:



CORE=MAX|*ssssss*

The amount of storage in bytes that the z/OS SORT/MERGE program can use for this application. The program defaults to CORE=MAX if no value is specified. When CORE=MAX is specified, DFSUCUM0 does not place a limit on the amount of storage the sort utility can use and uses the installation default value for the sort.

Recommendation: If you do not want to use the default specification of CORE=MAX and you are unsure about what to specify, use the minimum limit that is required for Data Facility Sort (DFSORT). If you are using an OEM sort product that allocates unit control blocks (UCBs) below the 16 MB line, use CORE=MAX.

DBRC
Can be specified as Y or N to override the specification of DBRC= in the installation defaults module DFSIDEF0. 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 is specified in the installation defaults module DFSIDEF0, 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 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.

DBRCGRP=*xxx*
Specifies the DBRC group that the utility should use. The variable *xxx*, a three digit alphanumeric value, must match the DBRC group ID specified in the RECON data set.

DD statements

The DFSUCUM0 utility uses a number of required and optional DD statements.

The following DD statements define the data sets that are used by this utility:

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 z/OS 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 z/OS SORT/MERGE program specifies AP (all messages to printer).

SORTLIB DD
Defines a data set containing load modules for the execution of the z/OS SORT/MERGE program. 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 z/OS 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 following DD statement must be used:

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

This data set can reside on a tape or a direct-access volume.

DFSUDD1 DD
Defines the new log output data set. The output data set can reside on a tape or a direct-access volume. The block size specified must be at least 64 and less than or equal to the device maximum (or 32760). If no block size is specified, the default block size is the device maximum. If the device is a 3380, a block size of 23476 is used. The logical record length and RECFM=VBS cannot be overridden by the DCB operand.

The DCB parameters specified within this program are RECFM=VBS and LRECL=32760. They cannot be modified. The output is blocked to the maximum device capacity. Standard labels must be used.

DFSUDD2 DD
Defines the new reformatted sort output log data set. It is used for diagnostic purposes only and is required if the SO control statement is specified. The output can reside on a tape or a direct-access volume and is blocked to either the device maximum or 32760, whichever is smaller. Standard labels must be used.

DFSULOG DD
Defines the log input data set containing the change records to be accumulated. This data set can reside on a tape or a direct-access volume.

Multiple log data sets can be used as input by concatenating the data sets. The log input must be in the sequence in which it was created. DBRC verifies that the log data sets are in chronological order, according to their START TIME.

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

RECON1 DD
Defines the first DBRC RECON data set and must be the same RECON data set defined for the control region.

If you are using dynamic allocation, do not include DD statements for the RECON data sets.

RECON2 DD
Defines the second DBRC RECON data set and must be the same RECON data set defined as second to the control region.

If you are using dynamic allocation, do not include DD statements for the RECON data sets.

RECON3 DD
Defines the third DBRC RECON data set and must be the same RECON data set defined as third to the control region.

If you are using dynamic allocation, do not include DD statements for the RECON data sets.

Return codes

The DFSUCUM0 utility return codes are:

Code	Meaning
0	All operations completed successfully.
4	Warning messages were issued.
8	One or more operations were not successful.
16	The z/OS SORT/MERGE program was not successful.

If the ERRET=ABEND option was specified when the z/OS SORT/MERGE program was installed, other return codes might be returned by the z/OS SORT/MERGE program.

These return codes can be tested by the COND= parameter on the EXEC statement of a subsequent job step.

Related concepts:

Initializing and maintaining the RECON data sets (System Administration)

Related tasks:

Using database change accumulation input for recovery (Database Administration)

Control statements for the DFSUCUM0 utility

The four utility control statements that are used by the Database Change Accumulation utility are: ID, DB0, DB1, and SO.

Except when certain parameters are specified, you can use all or any combination of these statements.

If you use the DBRC command GENJCL.CA to produce the JCL for the Database Change Accumulation utility, you do not need to code these control statements yourself and you are assured that the JCL specifications, including purge dates and times, are correct. If you do code the control statements yourself, be sure to specify DBRC=Y in the EXEC statement for the utility, so that DBRC verifies that the control statements are correct.

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. If you are running with DBRC support, the DFSUCUM0 utility applies the purge date provided by DBRC, otherwise no purge date is applied to the log records.
- 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: When including multiple control statements, the specification of *ALL or *OTHER in one statement can affect what you can specify in the other control statements. The following rules apply:

- If you include a DB0 statement, you cannot specify *ALL on any DB1 statement.
- If you specify *ALL on any DB0 statement, you cannot include any DB1 statements.
- If you specify *ALL on a DB1 statement, you cannot specify *OTHER on any other DB1 statements.

Recommendation: Use the DBRC command GENJCL.CA to generate the JCL for the Change Accumulation utility. Using the GENJCL.CA command ensures that the correct UTC purge time is applied and that the utility control statements are coded correctly for all change accumulation processing. However, if you code the JCL for the Change Accumulation utility manually, you are responsible for specify the correct purge date and time.

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 in DBRC or the utility control statement to the date and time in the previous change accumulation tapes. If the purge date is later than the date and time of a log record in the previous change accumulation tapes, 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.

Load database 1

Process - Log 1

Process - Log 2

Accumulate (Log 1, Log 2) into Accumulation Log-A

Process - Log 3

Process - Log 4

Reorganize database 1

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 database 1 was reorganized.

  Load database 2

  Process database 1 and database 2 - Log 7

  Process database 1 and database 2 - Log 8

  Reorganize database 2

  Accumulate (Log 7, Log 8) into Accumulation Log-C
- Purge date and time are now needed to eliminate previous log records for database 2 because database 2 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.

  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.

12-42 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. No offset defaults to the local offset.

43-68 ddnames

Positions 43 through 50, 52 through 59, and 61 through 68 can contain 1 to 3 ddnames which, combined with the database name supplied, make up the database data set identification.

The ddnames must be filled into the positions from left to right. 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.

69-80 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 and ddname combinations can be included.

Each combination of database name and 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.

12-42 Purge date and time. All records matching a record identification combination and dated before the purge date are eliminated.

43-68 ddnames

Positions 43 through 50, 52 through 59, and 61 through 68 can contain 1 to 3 ddnames which, combined with the database name supplied, make up the database data set 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.

69-80 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 statement 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 data set 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.

Related reference:

DBRC command syntax (Commands)

Examples for the DFSUCUM0 utility

Examples of how to use the DFSUCUM0 utility.

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 the following figure to the sample JCL:

Figure 59. DD statements for using DBRC without dynamic allocation

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

Subsections:

- "Accumulate database logs"
- "Accumulate all database change records" on page 209
- "Accumulate database logs without updating old data sets" on page 209

Accumulate database logs

In the following code sample, 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 2007 and before 1200 hours are to be eliminated. The second database (DI32DB02) is to be accumulated selectively. The change accumulation data set is recorded in RECON.

The database (DI32DB02) data set with the ddname DDI3IA is to be accumulated, and all records before day 173 of year 2007 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 2007 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,DBRC=Y'
//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 *
DB0DI32DB01071751200000
DB0DI32DB02071731500000                   DDI3IA
DB1DI32DB02071731500000                   DDI3OA
DB1 *OTHER
/*
```

Accumulate all database change records

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 2007 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   072000000000
/*
```

Accumulate database logs without updating old data sets

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 2007 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 2007 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 2007 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 *
DB0DI32DB01071751200
DB0DI32DB02071731500000                   DDI3IA
DB1DI32DB022007:173 15:15:30.2±8          DDI3OA
DB1 *OTHER
SO DI32DB01001 0000000100
/*
```

Chapter 16. Database Recovery utility (DFSURDB0)

Use the Database Recovery utility to recover a physically damaged data set in an IMS database. Only one data set is recovered for each execution.

This utility does not recover data from application logic errors; you must 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 batch region is defined as UDR on the PARM keyword of the utility EXEC statement.

The Database Recovery utility supports recovery in most operating environments, including:

- Remote Site Recovery (RSR) configurations
- IMS systems that use the integrated HALDB Online Reorganization function

At RSR tracking sites the Database Recovery utility recovers the databases to the same recovery point that the databases at the active site are recovered to.

If you are using RSR database level tracking, after the Database Recovery utility finishes running, the RSR function Online Forward Recovery completes the recovery process.

In systems that use the integrated HALDB Online Reorganization function, the Database Recovery utility can be used to recover both the input and the output data sets.

The following figure is a flow diagram of the Database Recovery utility.

© Copyright IBM Corp. 1974, 2015

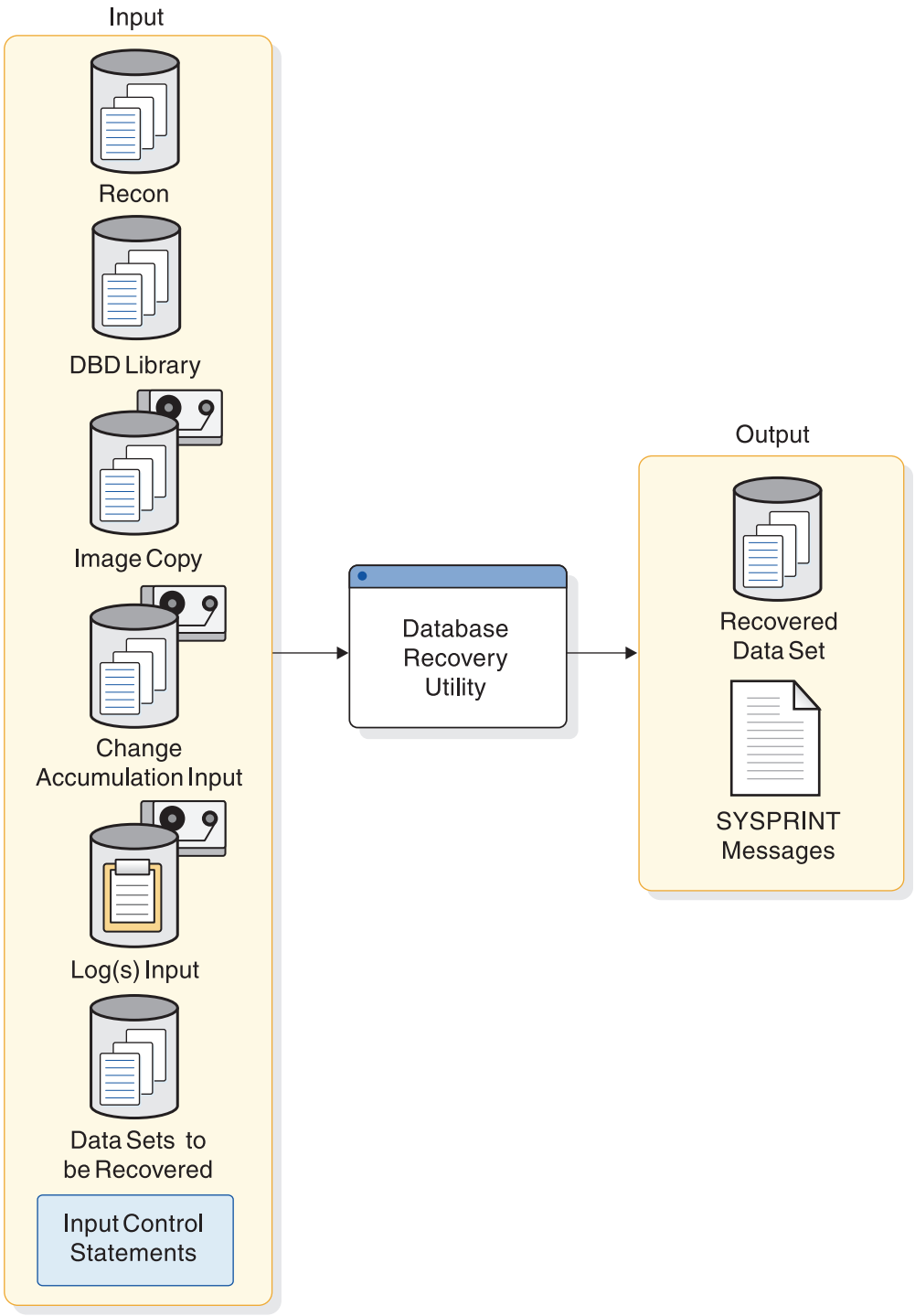


Figure 60. Database Recovery utility

Subsections:

- "Restrictions" on page 213
- "Prerequisites" on page 213
- "Requirements" on page 213
- "Recommendations" on page 215
- "Input and output" on page 215
- "Return codes" on page 220

• "JCL specifications" on page 218

Restrictions

The following restrictions apply when you run the Database Recovery utility:

- The Database Recovery utility cannot be used with HSAM or GSAM databases.
- Do not use data sets reloaded by the HD Reorganization Reload utility as a substitute for an image copy when performing a forward recovery. The physical location of segments might change during the reload, in which case the location of the segments would no longer match the location as recorded in the logs that the Database Recovery utility uses for recovery.
- Do not use output from the Log Merge utility as input to recovery.
- A nonstandard image copy data set cannot be used as input to the database recovery utility. The procedure for recovering a database with a nonstandard image copy is slightly different depending on whether the IMS system is an active or tracking subsystem.
- Utilities that alter databases cannot be run while the database is quiesced.

Prerequisites

If you want to use change accumulation data sets as input to the Database Recovery utility, run the Change Accumulation utility (DFSUCUM0) prior to executing the Database Recovery utility.

If you do not include image copies to the Database Recovery utility, you must restore the backup copy of the database prior to running the Database Recovery utility.

If you are recovering a DEDB multiple area data set (MADS), the MADS must be made unavailable and set to recovery needed. You can make a MADS unavailable by using the DBRC command CHANGE.ADS. For example, CHANGE.ADS ADDN(ADDN004) AREA(AREA002) DBD(DBD001) UNAVAIL

Requirements

The following requirements apply when you run the Database Recovery utility:

General requirements

- If concurrent image copy (CIC) is used, the minimum DBRC share level for CIC is 1. The share level must be specified to run CIC.
- 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.
- When recovering a shared secondary index, specify only the first index name listed in the DBDNAME parameter of the DBD statement of the shared secondary index. The Database Recovery utility recovers the data set for all indexes in the shared secondary index.
- If the HISAM unload data set will be used as the DFSUDUMP data set input to the Database Recovery Utility, the database must be reloaded immediately after it is unloaded. This procedure ensures that the Database Recovery Utility will not be affected by any database update log records that are created between HISAM unload and recovery operations. This procedure is necessary because the HISAM unload data set contains a reorganized form of the database. Log records that are

created between HISAM unload and reload operations reflect the physical state of the database before reorganization.

Attention: If log records that were created using a previous state of the database are used in combination with a HISAM unload data set, a data integrity exposure might be created because of the new state reflected in the unload data set. Reloading the data set after unloading it maps the data set to the newly reorganized state of the database found on the unload data set. All database log records that are created after a HISAM reload operation reflect the newly organized state of the database, and can safely be used by the Database Recovery Utility. If the change accumulation process is used, start a new job after reloading. Starting a new job eliminates exposure to any change accumulation records that were generated before the latest reload.

•

Fast Path DEDB requirements

- If a Fast Path DEDB area data set (ADS) is registered in with DBRC in the RECON data set, the following requirements apply:
  - The dataset1 DD statement that identifies the ADS being recovered by the Database Recovery utility must specify the same ddname and data set name that are registered in the RECON data set for the ADS.
  - The target ADS must be marked “recovery-needed” in the RECON.
- If you are recovering a Fast Path DEDB ADS that is not registered in the RECON data set and the RECON data set has a NOFORCE attribute, the ddname specified in the dataset1 DD statement must match the target area name.
- If you are recovering a Fast Path DEDB ADS that is not registered in the RECON data set and the RECON data set has a FORCER attribute, the Database Recovery utility terminates with an error message.
- If you are recovering a DEDB multiple area data set (MADS), you must use the DEDB Area Data Set Create utility to create the additional copies of the recovered MADS.

Data set requirements

- For full recovery of an OSAM data set, the OSAM data set must be deleted and reallocated prior to running recovery.

  The following example shows how to delete and reallocate an OSAM data set using IDCAMS:

```
IF MAXCC = 0                                        -
 THEN DO
    DELETE (IMSVS.IMSIVP.DFSIVD1) NONVSAM
    IF LASTCC NE 0                                  -
    THEN SET MAXCC = 0
    ELSE
    ALLOCATE                                        -
       DSNAME('IMSVS.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
```

Recommendation: Delete and allocate multi-volume OSAM data sets using the z/OS IDCAMS commands in a job step prior to the recovery job step. If they are not deleted and allocated, the data sets are allocated with DISP=NEW in the recovery job step. Any attempt to perform a full recovery using an existing OSAM data set containing data produces 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.
- For recovery of a VSAM data set intended to be restored from a fast replication copy made by Image Copy 2, the VSAM data set must be deleted and redefined as well 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.

Important: With the exception of forward recoveries when image copies are not included as input to the utility, the Database Recovery utility expects the input data set to be empty. If the data set is not empty, the recovery fails and IMS issues message DFS0792A.

Recommendations

Use DBRC when running the Database Recovery utility and generate your recovery JCL by using the DBRC GENJCL commands. DBRC performs the following actions when used with the Database Recovery utility:

- Generates the correct JCL for the utility when the GENJCL commands are used.
- Calculates the correct Universal Coordinated Time (UTC) offsets to use for the recovery. When DBRC is not used, then the z/OS UTC offset is used.
- Selects the correct image copy or data set to use as input for recovery.
- Selects the correct Change Accumulation dataset to use as input.
- Selects the correct logs to use as input to recovery.
- Computes the correct purge time to use for recovery.
- Validates all of the above input for correctness when the DBRC generated JCL executes.

When the database or area is quiesced, a DEALLOC time stamp is recorded in the allocation record for that database or area in the RECON data sets with the QUIESCE flag set and can be retrieved using the LIST.DB command, the LIST.DBDS command, or a QUERY API request. The utility can use this time stamp to recover the database.

Input and output

Input to the Database Recovery utility (DFSURDB0) can include image copies, change accumulation data sets, and logs. If you specify DBRC=Y on the utility EXEC statement, DBRC verifies that your JCL specifications for the utility input are correct. If DBRC determines that the specifications are incorrect, the Database Recovery utility terminates.

To be certain that all of your JCL specifications are correct, use the DBRC command GENJCL.RECOV to generate the JCL for the Database Recovery utility.

If you do not include an image copy as input to the Database Recovery utility, such as when you apply the image copy to the database in a prior job, the Database Recovery utility uses the data set to be recovered as input.

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.

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

The following table identifies inputs and outputs for the Database Recovery utility.

Table 11. Input to and output from the Database Recovery utility

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

Logs

Multiple logs can be provided by concatenating the logs in date and time sequence.

Only the following change records are valid input to database recovery:

- 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)

Image copies

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).
- An image copy created by either the concurrent copy function or the fast replication function of the Database Image Copy 2 Utility (DFSUDMT0). The database data set should be registered with DBRC if using an image copy created by DFSUDMT0.
- A HISAM unload data set created by the HISAM Reorganization Unload utility (DFSURUL0).

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

If you are using a nonstandard image copy of the database for recovery, such as an image copy created by using a z/OS copy utility, restore the database with the nonstandard image copy and notify DBRC of the nonstandard image copy before you run the Database Recovery utility. The steps for recovering from a nonstandard image copy differ depending on whether you are using a clean nonstandard image copy or a fuzzy nonstandard image copy.

If you are using a clean nonstandard image copy, use the following process for recovery:

1. Restore the database from the clean nonstandard image copy
2. Add information about the recovery to the RECON data set by issuing the DBRC command NOTIFY.RECOV
3. Specify DUMMY or NULLFILE in the DFSUDUMP DD statement when you either code the utility control statements or issue the DBRC command GENJCL.RECOV with the USEDBDS parameter
4. Apply the database changes from the log, the change accumulation data set, or both, to the restored nonstandard image copy by running the Database Recovery utility

If you are using a concurrent (or "fuzzy") nonstandard image copy, use the following process for recovery:

- Issue the CHANGE.DBDS command with the RECOV parameter to notify DBRC that a recovery is in progress
- Restore the database from the fuzzy nonstandard image copy
- If you are coding the Database Recovery utility control statements manually, specify the following:
  - DUMMY or NULLFILE in the DFSUDUMP DD statement
  - An M in column 64 to indicate that a concurrent user image copy was used for recovery. If performing a USER CIC, then recovery, M is specified in position 63, not 64
  - If a specific runtime is specified for recovery, a runtime specification control statement
- If you are generating the Database Recovery utility JCL by using the DBRC command GENJCL.RECOV, specify the following:
  - the DUMMY or NULLFILE in the DFSUDUMP DD statement
  - the USERIC parameter if you are specifying a specific runtime for recovery
  - the LASTUIC parameter if you are using the latest concurrent user image copy
- Apply the database changes from the log, the change accumulation data set, or both, to the restored nonstandard image copy by running the Database Recovery utility

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 data set 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 data set 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.

JCL specifications

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

- A JOB statement that you define to meet the specifications of your installation.
- 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
The name of the database (as defined in the DBD statement) that includes the data set to be recovered. For HALDB databases, this is the name of the master database.

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

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, the Database Image Copy 2 utility, or the HISAM Reorganization Unload utility. The data set can reside on a tape or a direct-access volume.

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.

This DD statement is DUMMY if any of the following conditions are true:

- no image copy or HISAM unload copy input is supplied
- a HALDB Online Reorganization is used as the starting point for recovery
- the USEDBDS or USEAREA keyword is specified on the GENJCL.RECOV command

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.

DFSUSNAP DD
Defines a SNAP output data set that is used to write specific areas of storage for diagnostic purposes where appropriate. The data set can be defined as SYSOUT=* or with the following attributes:
DCB=(RECFM=VBA,LRECL=125,BLKSIZE=1632)

dataset1 **DD**
Defines the data set to be recovered. The ddname must be the same as the name that describes this data set in the DBD. If a dataset1 DD statement is included, the value represented by dataset1 must also match the ddname specified in columns 13–20 of the DFSURDB0 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, SMSCIC, SMSONLC, or SMSOFFLC image copy created by the Database Image Copy 2 utility is used as 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

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.

Return codes

The Database Recovery utility provides the following return codes:

Code	Meaning
0	All operations completed successfully.
4	Warning messages were issued.
8	More than one operation was not successful.
16	Severe errors caused the job to terminate before completing all operations.

These return codes can be tested by the COND= parameter on the EXEC statement of a subsequent job step.

Related concepts:

IMS buffer pools (System Definition)

Understanding the operations task (System Administration)

Assigning a sharing level with DBRC (System Administration)

Related tasks:

Planning your database recovery strategy (Database Administration)

Control statements for the DFSURDB0 utility

The Database Recovery utility uses two utility control statements: ABEND and NOSEQCK. These statements can be used separately or together.

Subsections:

- "ABEND statement"
- "NOSEQCK statement"
- "Database Recovery statement"
- "Runtime specification statement for non-standard concurrent image copies" on page 223

ABEND statement

Use the ABEND 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 the NOSEQCK utility control statement to request that the utility not perform sequence checking on the input logs.

Attention: Use this statement with 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.

Database Recovery statement

The Database Recovery utility control statement has a fixed format using the following field positions:

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. For HALDB databases, the dbdname is the name of

the partition that contains the data set being recovered. If you enter a DBD name in the PARM field of the EXEC statement, the DBD name must match the DBD name that is in the Database Recovery utility control 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 *dataset1* DD statement.

  DBRC must be active in order to specify a data set name as input to recovery.

  For a recovery from an image copy other than a fast replication image copy, the JCL stream must contain a *dataset1* DD statement, corresponding to the DD name for the data set, on which you must code the data set name of the DBDS to be recovered.

  For a recovery from a fast replication image copy, the corresponding *dataset 1* DD statement is optional. If the *dataset1* DD statement is included, the target data set or area name is provided by the DD statement. If it is not included, the target data set or area name is provided by DBRC.

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

31-61 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
31-47	This position specifies the date and time, in the format yydddhhmmssthmiju, where yy=year, ddd=day of year, hh=hour, mm=minute, ss=second, thmiju=millionths of a second.
48	This position specifies the sign of the offset, + or -.
49-52	This position specifies the offset from the UTC, in the format HHMM.
53-61	Blanks

Or alternatively:

Position	Description
31-52	This position specifies the punctuated time stamp, in the format yy.ddd hh:mm:ss.thmiju, where yy=year, ddd=day of year, hh=hour, mm=minute, ss=second, thmiju=millionths of a second.
53	Blank
54	This position specifies the sign of the offset, + or -.

55-59 This position specifies the offset from the UTC, in the format HH:MM.

60-61 Blanks

Or alternatively:

Position
Description

31-54 This position specifies the punctuated time stamp with a four-year digit, in the format yyyy.ddd hh:mm:ss.thmiju, where yyyy=year, ddd=day of year, hh=hour, mm=minute, ss=second, thmiju=millionths of a second.

55 Blank

56 This position specifies the sign of the offset, + or -.

57-61 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).

62 blank

63 Recovery type. Valid values are:

C Specifies that an image copy data set is not being used for this recovery.

V Specifies a RECEIVE type recovery is being performed at an RSR tracking site

M Specifies a user concurrent image copy is being used for recovery

64 Specifies how the data set information is provided. The valid values are:

D Specifies that the fully-qualified data set name is provided by DBRC and that a DD statement is not required in the job stream. The D specification is equivalent to the NODBDSDD parameter of the GENJCL.RECOV command.

blank
Specifies that a DD statement for the DBDS is included in the generated JCL. This parameter is the default. If position 64 is left blank, it is equivalent to specifying the DBDSDD parameter of the GENJCL.RECOV command.

65–80 Available for user comments.

Runtime specification statement for non-standard concurrent image copies

To specify a specific run time when recovering from a user concurrent image copy, use the runtime specification control statement.

The runtime specification control statement is identified by entering an “M” in position 1, followed by the run time in positions 2–32.

Examples for the DFSURDB0 utility

The following examples show sample JCL for the DFSURDB0 utility.

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 the following figure to the sample JCL:

Figure 61. DD statements for using DBRC without dynamic allocation

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

Subsections:

- "Recover HDAM OSAM data set JCL"
- "Recover a DEDB" on page 225
- "Recover a partition data set in a database with multiple partitions" on page 225
- "Recovery with fast replication a fully-qualified data set name" on page 225
- "Recovery with a concurrent user image copy" on page 226
- "Recovery with a time stamp" on page 227

Recover HDAM OSAM data set JCL

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
/*
```

Recover a DEDB

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
/*
```

Recover a partition data set in a database with multiple partitions

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.

```
//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
//RECON3   DD DSN=IMSTESTL.IMS.RECON3,DISP=SHR
//DFSVSAMP DD input for VSAM and OSAM buffers and options
//SYSIN DD *
S  MASTERDB PART001A
/*
```

Recovery with fast replication a fully-qualified data set name

This example shows the recovery JCL when fast replication is used and a fully-qualified data set name is provided by DBRC, as indicated by a D in column

64 of the Recovery utility control statement. A DD statement is not required in the job stream. The D specification is equivalent to the NODBDSDD parameter of the GENJCL.RECOV command.

```
//RECOVERY JOB
//*                                                                     00150000
//STEP1 EXEC PGM=DFSRRC00,PARM='UDR,DFSURDB0,DXVNTZ02'                  00120000
//*                                                                     00150000
//STEPLIB   DD DSN=IMS.SDFSRESL,DISP=SHR                                00650000
//DFSRESLB  DD DSN=IMS.SDFSRESL,DISP=SHR                                00650000
//IMS       DD DSN=IMS.DBDLIB,DISP=SHR                                  00650000
//SYSPRINT  DD SYSOUT=A                                                 00500000
//RECON1    DD DSN=IMS.RECON1,DISP=SHR
//RECON2    DD DSN=IMS.RECON2,DISP=SHR
//RECON3    DD DSN=IMS.RECON3,DISP=SHR
//DFSUDUMP  DD DSN=IMS.DXVNTZ02.XDLBT04I.IC,                            00950000
//             DISP=(OLD,KEEP),DCB=BUFNO=10                             01100000
//DFSVDUMP  DD DUMMY                                                    01133000
//DFSUCUM   DD DUMMY                                                    01340000
//DFSULOG   DD DSN=IMSLOG.MONDAY,DISP=(OLD,KEEP),                       01400000
//             UNIT=TAPE,VOL=SER=LOG1,LABEL=(1,SL),                     01450000
//             DCB=RECFM=VB,                                            01500100
//             DISP=OLD                                                 01550000
//DFSVSAMP  DD *                                                        01900000
1024,4
4096,4
8192,4
//*-+----1----+----2----+----3----+----4----+----5----+----6----+----7--

//SYSIN     DD *                                                        02050000
S  DXVNTZ02 XDLBT04I                                        D           02070000
/*
```

Recovery with a concurrent user image copy

This example shows the recovery JCL when a concurrent user image copy (CIC) is used for recovery, as indicated by an M in column 63 of the database recovery statement. Following the database recovery statement is the runtime statement that specifies the runtime of the concurrent user image copy.

```
//RECOVER   JOB
//*                                                                     00150000
//STEP1 EXEC PGM=DFSRRC00,PARM='UDR,DFSURDB0,DHVNTZ02'                  00120000
//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
//RECON1    DD DSN=IMS.RECON1,DISP=SHR                                  00550000
//RECON2    DD DSN=IMS.RECON2,DISP=SHR                                  00600000
//RECON3    DD DSN=IMS.RECON3,DISP=SHR                                  00620000
//HIDAM    DD DSN=IMSTESTL.DHVNTZ02.FKXXI01E,                           00700000
//             DISP=OLD                                                 00920000
//DFSUDUMP  DD DUMMY                                                    01120000
//DFSVDUMP  DD DUMMY                                                    01133000
//DFSUCUM   DD DUMMY                                                    01340000
//DFSULOG   DD DSN=IMSLOG.MONDAY,DISP=(OLD,KEEP),                       01400000
//             UNIT=TAPE,VOL=SER=LOG1,LABEL=(1,SL),                     01450000
//             DCB=RECFM=VB,                                            01500000
//             DISP=OLD                                                 01500100
//DFSVSAMP  DD *                                                        01900000
1024,4                                                                  01950000
4096,4                                                                  02000000
8192,4                                                                  02000000
//SYSABEND DD SYSOUT=A                                                  02050000
//SYSUDUMP DD SYSOUT=A                                                  02050000
//SYSIN     DD *                                                        02050000
```

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

S  DHVNTZ02 HIDAM                                              M         02070000
M062501219001-0700                                                        02070100
/*                                                                        03000000
```

Recovery with a time stamp

This example shows the recovery JCL for a time stamp recovery. The time indicated in column 31 corresponds to the time input by using the RCVTIME parameter of the DBRC command GENJCL.RECOV.

Remember: A time stamp of a HALDB partition requires the recovery of all partitions in the HALDB database and the subsequent rebuilding of any associated primary indexes any indirect list data sets (ILDS).

```
//RECOVERY  JOB
//*                                                                       00150000
//RCV1 EXEC PGM=DFSRRC00,REGION=0M,PARM='UDR,DFSURDB0,DIVNTZ02'           00120000
//*                                                                       00150000
//STEPLIB   DD DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB  DD DSN=IMS.SDFSRESL,DISP=SHR
//IMS       DD DSN=IMS.DBDLIB,DISP=SHR                                    00650000
//SYSPRINT  DD SYSOUT=A                                                   00500000
//RECON1    DD DSN=IMS.RECON1,DISP=SHR                                    00550000
//RECON2    DD DSN=IMS.RECON2,DISP=SHR                                    00600000
//RECON3    DD DSN=IMS.RECON3,DISP=SHR                                    00620000
//DBHVSAM1    DD DSN=IMSTESTL.DIVNTZ02.FJXXS01K,                          00700000
//             DISP=OLD                                                   00920000
//DFSUDUMP  DD DSN=IMS.DIVNTZ02.DBHVSAM1.IC,                              00950000
//             UNIT=SYSDA,                                                00980000
//             VOL=(PRIVATE,,,,SER=(222222)),                             01000000
//             LABEL=(1,SL),                                              01050000
//             DISP=(OLD,KEEP),DCB=BUFNO=10                               01100000
//DFSVDUMP  DD DUMMY                                                      01133000
//DFSUCUM   DD DUMMY                                                      01340000
//DFSULOG   DD DUMMY                                                      01670000
//DFSVSAMP  DD *                                                          01900000
1024,4                                                                    01950000
4096,4                                                                    02000000
8192,4                                                                    02000000
//SYSABEND DD SYSOUT=A                                                    02050000
//SYSUDUMP DD SYSOUT=A                                                    02050000
//SYSIN     DD *                                                          02050000
//*-+----1----+----2----+----3----+----4----+----5----+----6----+----7--

S  DIVNTZ02 DBHVSAM1          2006.250 12:12:49.123456 -07:00           02070000
/*                                                                        03000000
```

Related reference:

Chapter 19, “HALDB Index/ILDS Rebuild utility (DFSPREC0),” on page 241

Chapter 17. DEDB Area Data Set Compare utility (DBFUMMH0)

Use the DEDB Area Data Set Compare utility (DBFUMMH0) to check for errors in the physical records of a multiple area data set (MADS) by comparing all area data set copies of the MADS.

The DBFUMMH0 utility compares the control intervals (CIs) of:

- The root addressable part and the independent overflow part
- A 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.

Subsections:

- "Restrictions"
- "Prerequisites" on page 230
- "Requirements" on page 230
- "Recommendations" on page 230
- "Input and output" on page 230
- "JCL specifications" on page 231
- "Return codes" on page 231

Restrictions

To run the DBFUMMH0 utility you must execute the DBFUMMH0 utility in a separate job step because the program cannot switch from one utility or one database to another within the same job step.

The DBFUMMH0 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 DBFUMMH0 utility terminates if:

- The to-be-compared area data set or area is stopped by a /STOP AREA or /STOP ADS command

© Copyright IBM Corp. 1974, 2015

- 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

The DBFUMMH0 utility cannot be restarted. If the restart (REST) parameter is used, it is ignored.

The DBFUMMH0 utility will work on Fast Path (DEDB) databases only.

Utilities that alter databases cannot be run while the database is quiesced.

Prerequisites

Currently, no prerequisites are documented for the use and execution of the DBFUMMH0 utility.

Requirements

Currently, no requirements are documented for the DBFUMMH0 utility.

Recommendations

Currently, no recommendations are documented for the use and execution of the DBFUMMH0 utility.

Input and output

The DBFUMMH0 utility accepts as input a data set that contains the input parameters supplied by DEDB online utility commands.

The DBFUMMH0 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

Example of an error queue element (EQE) status report

The following example 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 of a printed dump of unmatched records

The following example 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
```

JCL specifications

The DBFUMMH0 utility is executed as a standard z/OS job. The JCL specifications for the DBFUMMH0 utility include a JOB statement, the EXEC statement, and the DD statements. To control the processing options of the DBFUMMH0 utility you must code DEDB online utility commands in the SYSIN DD statement.

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 it can be in the form PGM=DFSRRC00.

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.

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

Return codes

Currently, no return codes are documented for the DBFUMMH0 utility.

Related reference:

FPUTIL procedure (System Definition)

Control statements for the DBFUMMH0 utility

The DBFUMMH0 utility control statements are submitted in the form of DEDB online utility commands. The control statements are coded in the SYSIN DD statement. The control statements are also used to describe the input area data sets that are being compared.

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.

AREA (required)
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 - 8 characters. The first character must be alphabetic, and the rest of the characters are alphanumeric. Alphabetic characters include @, #, and $.

BUFNO (optional)
Specifies the number of buffers to use to read or write the DEDB.

For the utilities other than the Reorganization utility, a minimum of seven 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.

DDNAME (optional)
Specifies the area data set to be compared. 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.

TYPE (required)
Specifies either the compare, or create utility as the type of run.

Attention: 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.

Related reference:
"Running the DBFUHDR0 utility" on page 315

Examples for the DBFUMMH0 utility

These examples provide sample JCL for comparing one, seven, or all available area data sets.

Subsections:

- "Compare two area data sets"
- "Compare seven area data sets"
- "Compare all available area data sets" on page 234

Compare two area data sets

The following example 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
/*
```

Compare seven area data sets

The following example 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
/*
```

Compare all available area data sets

The following example 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
/*
```

Chapter 18. 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.

You can also use the DBFUMRI0 utility to create additional copies of a Virtual Storage Option (VSO) areas. However the need for multiple copies of a VSO area is less than it is for non-VSO areas.

The need for multiple area data sets of VSO areas is reduced because after 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.

The performance of the DBFUMRI0 utility when creating copies of a VSO area 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 VSO area 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 a VSO area is not kept in the z/OS data space, so the DBFUMRI0 utility must still read SDEPs from DASD to copy them to a new area data set.

VSO areas do not affect the write function of the DBFUMRI0 utility.

If the DBFUMRI0 utility detects a read error in a CI of an available data set, the utility uses another data set in the same area. If all the available data sets have read errors in the same CI, the DBFUMRI0 utility terminates.

The DBFUMRI0 utility also terminates if it detects a write error in the new data set.

You can stop the DBFUMRI0 utility before it completes by issuing the /STOP REGION command.

Tip: The area format phase of the DBFUMRI0 utility copy process can cause delays to XRF takeovers and to execution of the /STOP REGION command. IMS

© Copyright IBM Corp. 1974, 2015

suspends both XRF takeovers and the /STOP REGION command until the area format stage completes. You can avoid this delay by preformatting the area data sets.

Subsections:

- "Restrictions"
- "Prerequisites"
- "Requirements"
- "Recommendations"
- "Input and output" on page 237
- "Return codes" on page 237
- "JCL specifications" on page 237

Restrictions

The following restrictions apply when using the DEDB Area Data Set Create utility:

- The DBFUMRI0 utility cannot be restarted. It must be rerun from the beginning.
- If the DBFUMRI0 utility is running in an XRF active IMS system in a data sharing environment, the takeover process is suspended until the format phase ends.
- The DBFUMRI0 utility will work on Fast Path (DEDB) databases only.
- Utilities that alter databases cannot be run while the database is quiesced.

Prerequisites

Currently, no prerequisite tasks are documented for the DBFUMRI0 utility.

Requirements

To run the DBFUMRI0 utility you must satisfy the following operational requirements:

- You must execute the DBFUMRI0 utility in a separate job step. The program cannot switch from one utility or one database to another within the same job step.
- The size of the CI must be identical to that defined for the other data sets of the same area.

  The data set is defined by the VSAM definition. 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.

Recommendations

When running the DBFUMRI0 utility in XRF environments, to prevent delays in the event of an XRF takeover, preformat the new ADS by using the DEDB Initialization utility (DBFUMIN0) with DBRC=N specified. IMS suspends XRF takeovers, as well as the execution of the /STOP REGION command, until the format phase of the DBFUMRI0 utility completes.

When area data sets are preformatted, the DBFUMRI0 utility skips the format phase.

When running the DBFUMIN0 utility to preformat area data sets for the DBFUMRI0 utility, DBRC=N 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 complete.

Input and output

The DEDB Area Data Set Create utility uses the following input:

- The area data set to be copied
- DBRC RECON data set
- A data set that contains the input parameters supplied by 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

Return codes

Currently, no return codes are documented for the DBFUMRI0 utility.

JCL specifications

The DBFUMRI0 utility is executed as a standard z/OS job. The JCL specifications for the DBFUMRI0 utility include a JOB statement, the EXEC statement, and the DD statements. To control the processing options of the DBFUMRI0 utility you must code DEDB online utility commands in the SYSIN DD statement.

EXEC statement

This statement can either specify the FPUTIL procedure which contains the required JCL or be in the form:

```
PGM=DFSRRC00
```

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.

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

Control statements for the DBFUMRI0 utility

The DBFUMRI0 utility control statements are submitted in the form of DEDB online utility commands. The commands are coded in the SYSIN DD statement. The control commands also describe the input area data sets that are being created.

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.

AREA (required)
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 - 8 characters. The first character must be alphabetic, and the rest of the characters are alphanumeric. Alphabetic characters include @, #, and $.

BUFNO (optional)
Specifies the number of buffers to use to read or write the DEDB.

For the utilities other than the Reorganization utility, a minimum of seven 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.

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 "Buffers and the BUFNO command" on page 315 for information on how BUFNO is used for the High-Speed DEDB Direct Reorganization utility.

DDNAME (required)
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.

GO (optional)
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.

TYPE (required)
The CREATE operator is required for the CREATE utility.

Attention: 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.

Examples for the DBFUMRI0 utility

These examples provide sample JCL for creating one, two, or five new area data sets.

Subsections:

- "JCL for creating one new area data set"
- "JCL for creating two new area data sets" on page 240
- "JCL for creating five new area data sets" on page 240

JCL for creating one new area data set

The following example 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
/*
```

JCL for creating two new area data sets

The following example 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
/*
```

JCL for creating five new area data sets

The following example 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 19. HALDB Index/ILDS Rebuild utility (DFSPREC0)

Use the HALDB Index/ILDS Rebuild utility (DFSPREC0) to rebuild the primary index data set of a PHIDAM database and the indirect list data set (ILDS) of either a PHDAM or a PHIDAM database after a partition recovery or an offline partition reorganization that did not update the ILDS.

The DFSPREC0 utility scans the HALDB partition database data sets (DBDSs) and recreates the ILDS and the PHIDAM primary index from the HALDB partition data. A single execution of the DFSPREC0 utility can rebuild the primary index only, the ILDS only, or both for the target partition.

The HALDB Index/ILDS Rebuild utility runs as a batch utility in a Type=ULU region.

The DFSPREC0 utility processes only one partition at a time. This allows you to run the DFSPREC0 utility against multiple partitions concurrently, which greatly reduces the amount of time that is needed to rebuild the primary indexes and ILDS of multiple partitions in a HALDB database. The DFSPREC0 utility can rebuild both the primary index and the ILDS for a HALDB partition during a single execution of the utility.

The DFSPREC0 utility can rebuild the ILDS by using either VSAM update mode or VSAM load mode, depending on the recovery type you specify in the utility control statement. If you use VSAM load mode the specifications for free space that you make in the VSAM DEFINE statement are applied to the ILDS, which both reduces CI and CA splits and improves performance.

The number of index entries in an ILDS can change significantly when it is rebuilt, if the ILDS contained a large number of ILEs for segments that no longer exist in the database. Unused ILEs are removed from the rebuilt ILDS.

The DFSPREC0 utility supports the integrated HALDB Online Reorganization function and can rebuild primary indexes and an ILDSs when the reorganization function is stopped, but is still in a cursor-active state.

Subsections:

- "Restrictions"
- "Prerequisites" on page 242
- "Requirements" on page 242
- "Input and output" on page 242
- "JCL specifications" on page 242
- "Return codes" on page 244

Restrictions

The DFSPREC0 utility works on full-function HALDB databases only.

Utilities that alter databases cannot be run while the database is quiesced.

© Copyright IBM Corp. 1974, 2015

The PHIDAM index data set or ILDS must be deleted and redefined before it is rebuilt.

Prerequisites

If you need to recover the database data sets of a partition in addition to rebuilding the primary index and ILDS, recover the database data sets before running the Index/ILDS Rebuild utility. The Index/ILDS Rebuild utility reads the partition data sets as it rebuilds the primary index and ILDS. Otherwise, you are not required to recover the input partition prior to running the Index/ILDS Rebuild utility.

If you need to rebuild both the primary index and the ILDS of a PHIDAM partition, rebuild the primary index first and then rebuild the ILDS. For PHIDAM partitions, the Index/ILDS Rebuild utility uses the primary index to rebuild the ILDS. If you rebuild both the primary index and the ILDS in a single execution of the Index/ILDS Rebuild utility by specifying either BOTH or BOTHF in the utility control statement, the Index/ILDS Rebuild utility automatically rebuilds the primary index first.

Requirements

- If you are using the VSAM load mode option of the Index/ILDS Rebuild utility, as specified by either the ILEF or the BOTHF parameter in the utility control statement, the utility must have five z/OS data spaces available to it for processing.
-

Input and output

The HALDB Index/ILDS Rebuild utility uses a control statement. The control statement allows you to specify the partition name and the type of recovery that you want the DFSPREC0 utility to perform.

Output from the HALDB Index/ILDS Rebuild utility includes the rebuilt index, rebuilt ILDS, and a report of the number of entries that are inserted into the PHIDAM primary index and into the ILDS. The report is written to the SYSPRINT data set, which also contains any error messages that are returned by the HALDB Index/ILDS Rebuild utility.

The HALDB Index/ILDS Rebuild utility produces a rebuilt PHIDAM index, a rebuilt ILDS, or both. The DFSPREC0 utility also reports the total number of lines that are sent to the SYSPRINT data set, which indicates the number of index and ILE entries created.

JCL specifications

The following JCL statements are required:

- A JOB statement that you define to meet the specifications of your installation.
- An EXEC statement
- DD statements that specify the input and output

You must also include a utility control statement.

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 that make up this HALDB will be rebuilt in the execution of the DFSPREC0 utility. 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.

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.

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 that is required by the DL/I Buffer Handler.

Return codes

The return codes that are generated by the Index/ILDS Rebuild utility are returned in message DFS1982I.

Related concepts:

Using IMS utilities with HALDB Online Reorganization (Database Administration)

Related reference:

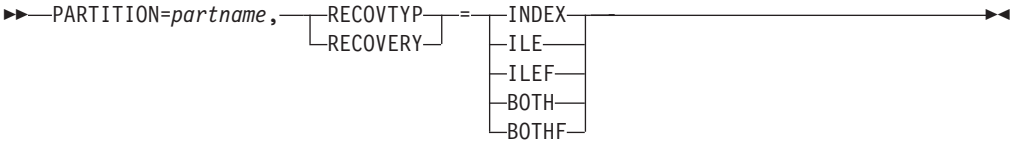
DBBBATCH procedure (System Definition)

DLIBATCH procedure (System Definition)

Control statement for the DFSPREC0 utility

Use the utility control statements to describe processing options for the HALDB Index/ILDS Rebuild utility.

The HALDB Index/ILDS Rebuild utility control statement uses a fixed-position format and can include the parameters shown in the following syntax diagram. The parameter positions are described immediately following the syntax diagram.



The keywords and parameters of the DFSPREC0 utility control statement must be entered in the following fixed positions:

Position
Description

1–10 Fixed character keyword. This must be characters PARTITION=.

11–17 Partition name. Name of the HALDB partition containing the primary index data set, the ILDS, or both, to be rebuilt.

18 Comma.

19–27 Fixed character keyword. This must be either the RECOVTYP or RECOVERY.

28-32 Recovery type. Specify one of the following character values:

BOTH
Rebuilds both the primary index and the ILDS of the specified partition.

BOTHF
Rebuilds both the primary index and the ILDS of the specified partition. Rebuilds the ILDS by using the free space option. The utility must have five z/OS data spaces available for processing if you select the free space option.

ILE
Rebuilds only the ILDS of the specified partition.

ILEF
Rebuilds only the ILDS of the specified partition by using the free space option. DFSPREC0 must have five z/OS data spaces available for processing if you select the free space option.

INDEX
Rebuilds only the primary index of the specified partition.

Examples for the DFSPREC0 utility

The sample JCL for DFSPREC0 in the following subtopics rebuilds both the primary index and the ILDS for partition PARTNAM.

The first sample uses the VSAM update mode to rebuild the ILDS. The second sample uses the free space option to rebuild the ILDS.

Subsections:

- "VSAM update mode JCL example"
- "VSAM load mode JCL example"

VSAM update mode JCL example

The following sample JCL rebuilds both ILDS and primary index. In the example, PARTNAM 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.SDFSRESL,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
/*
```

VSAM load mode JCL example

The following sample JCL uses the free space option when rebuilding both ILDS and primary index. In the example, PARTNAM 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.SDFSRESL,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=BOTHF
/*
```

Running the DFSPREC0 utility

You can run the DFSPREC0 utility by using either the update option, which uses VSAM update mode, or the free space option, which uses VSAM load mode.

The differences between rebuilding an ILDS in VSAM update mode and VSAM load mode are:

VSAM update mode

If you specify ILE or BOTH in the utility control statement, the DFSPREC0 utility reads the partition data sets sequentially and updates each indirect list entry (ILE) in the ILDS immediately upon reading the target segment of the ILE. The DFSPREC0 utility updates the ILDS by using VSAM update mode. When the DFSPREC0 utility finishes reading the partition data sets, it is also finished rebuilding the ILDS. VSAM update mode does not add or maintain free space in the ILDS.

Use VSAM update mode:

- To retain the ILEs for deleted segments for recovery purposes
- To update an ILDS that requires only minimal updates

VSAM load mode

If you specify ILEF or BOTHF in the utility control statement, the DFSPREC0 utility reads the partition data sets sequentially, but instead of updating the ILEs immediately, the DFSPREC0 utility stores the ILEs in one of five data spaces. When the DFSPREC0 utility finishes reading the partition data sets, it sorts the ILEs by their indirect list keys (ILKs) and then loads all of the ILEs into the ILDS in a single pass by using VSAM load mode. Each ILE is loaded with the amount of free space specified in the VSAM KSDS DD statement that defines the ILDS, which is why this method is referred to as the *free space option.*

The free space option requires that the DFSPREC0 utility has five z/OS data spaces available to it for processing.

The free space option also requires an empty VSAM ILDS data set when running the DFSPREC0 utility. A job deleting and redefining the ILDS data set should be run prior to executing the DFSPREC0 utility.

Use VSAM load mode:

- To add free space to the ILDS for improved performance
- To rebuild an ILDS that requires a significant amount of updates
- To eliminate CI and CA splits in the ILDS
- To purge the ILDS of ILEs for segments that no longer exist in the active partition

When an ILDS requires heavy updating, the free space option can be faster than the VSAM update mode. Also, eliminating CI and CA splits, as well as adding free space to prevent splits in the future, can improve the performance of subsequent reorganizations and recoveries of HALDB partitions.

Chapter 20. MSDB Dump Recovery utility (DBFDBDR0)

Use the MSDB Dump Recovery utility (DBFDBDR0) to create a new copy of an MSDBINIT data set, or create a new MSDBINIT data set with one or more selected MSDBs, or reconstruct MSDBs when an IMS system terminates abnormally and emergency restart is unable to recover the MSDBs

To create new MSDBINIT data sets, the DBFDBDR0 utility uses either a dump data set (MSDBDUMP) or checkpoint data sets (MSDBCP1 and MSDBCP2). In an XRF environment, the checkpoint data sets can be either MSDBCP1 and MSDBCP2 or MSDBCP3 and MSDBCP4.

To reconstruct MSDBs when the IMS system terminates abnormally and emergency restart is unable to recover the MSDBs, run the DBFDBDR0 utility in RECOVERY mode. The DBFDBDR0 utility reconstructs the MSDBs and creates a new copy of the MSDBINIT data set by using the checkpoint data sets and the associated system log.

The MSDBs are reconstructed on the MSDBINIT data set by applying all of the logged changes to the MSDBDUMP data set.

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, you can force recovery from the MSDBDUMP data set by specifying that data set in both checkpoint DD statements.

To create a new MSDBINIT with one or more selected MSDBs, run the DBFDBDR0 utility in UNLOAD mode with the MSDBDUMP data set as input. The MSDBs are unloaded onto the MSDBINIT data set at the same level of change as they were dumped.

The DBFDBDR0 utility is an offline utility that runs in a z/OS batch region.

The following figure shows a flow diagram of the data sets used by the MSDB Dump Recovery utility for unloading and reconstructing MSDBs.

© Copyright IBM Corp. 1974, 2015

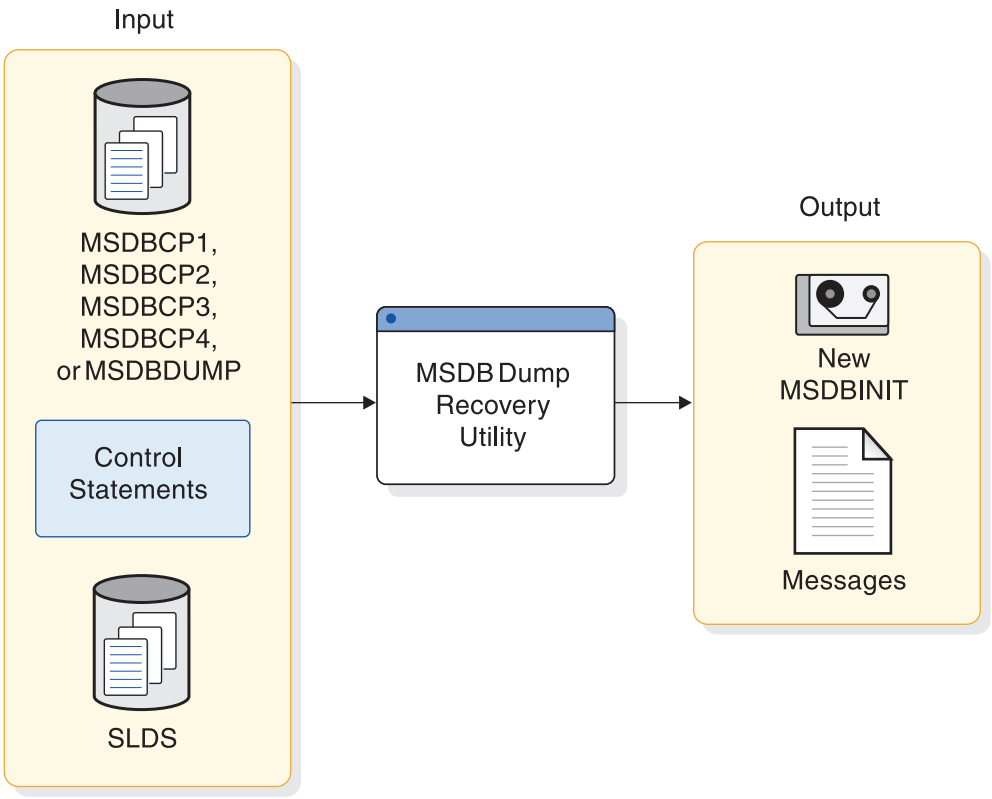


Figure 62. MSDB Dump Recovery utility input and output data sets

Subsections:

- "Restrictions"
- "Prerequisites"
- "Requirements"
- "Recommendations" on page 249
- "Input and output" on page 249
- "JCL specifications" on page 250
- "Return codes" on page 251

Restrictions

The DBFDBDR0 utility will work on Main Storage (MSDB) databases only.

Utilities that alter databases cannot be run while the database is quiesced..

Prerequisites

Currently, no prerequisites are documented for the DBFDBDR0 utility.

Requirements

The DBFDBDR0 utility requires at least 512KB of virtual storage.

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.

To unload MSDBs:

- The MSDBDUMP data set must be supplied.
- The control data set must specify the UNLOAD control statement.

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.
- You can supply the IMS log data set containing MSDB changes logged since the older checkpoint data set was created.
- The control data set must specify the RECOVERY control statement.

Recommendations

Currently, no recommendations are documented for the DBFDBDR0 utility.

Input and output

The primary input to the DBFDBDR0 utility is either an MSDB dump data set or MSDB checkpoint data sets. The primary output of the DBFDBDR0 utility is a new MSDBINIT. Output also includes a message data set.

The following table identifies inputs and outputs for the MSDB Dump Recovery utility.

Table 12. Data set requirements for the MSDB Dump Recovery utility

Input	Output
MSDBCP1, MSDBCP2, MSDBCP3, MSDBCP4, or MSDBDUMP	New MSDBINIT
Control statements	Messages
SLDS	

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).

The primary output is a new MSDBINIT data set. A message data set is also produced.

JCL specifications

The DBFDBDR0 utility is executed as a standard z/OS job. The JCL specifications for the DBFDBDR0 utility include a JOB statement, the EXEC statement, and the DD statements. One or more utility control statements must be/can be included with the JCL statements.

EXEC statement

The EXEC statement for the DBFDBDR0 utility can either invoke a cataloged procedure containing the required statements or be specified in the JCL by using the following format:

```
PGM=DBFDBDR0
```

DD statements

The DBFDBDR0 utility uses a number of required and optional 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 data set 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.

Return codes

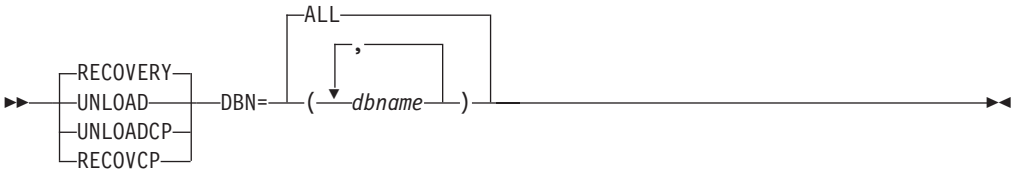
The DBFDBDR0 utility generates the following return codes:

Code	Meaning
0	Utility executed successfully.
4	Error message printed.
8	Unable to open print data set.
12	I/O error in print routine.

Control statements for the DBFDBDR0 utility

The control statements for the DBFDBDR0 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.



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.

Examples for the DBFDBDR0 utility

These examples show the JCL and utility control statements needed to execute the DBFDBDR0 utility.

Subsections:

- "MSDB unload"
- "Reconstruct MSDBs from older checkpoint data sets"
- "Unload a particular MSDB from the MSDBDUMP data set" on page 253
- "Reconstruct one MSDB with checkpoint data set and the system log" on page 253
- "Reconstruct MSDB from checkpoint data set of the later pair and system log" on page 253
- "Recreate new MSDBCP1 data set with RECOVCP utility control statement" on page 254

MSDB unload

This example 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
/*
```

Reconstruct MSDBs from older checkpoint data sets

This example 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
/*
```

Unload a particular MSDB from the MSDBDUMP data set

This example unloads one 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)
/*
```

Reconstruct one MSDB with checkpoint data set and the system log

The following example reconstructs one 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)
/*
```

Reconstruct MSDB from checkpoint data set of the later pair and system log

The following example 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)
/*
```

Recreate new MSDBCP1 data set with RECOVCP utility control statement

The following example 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
/*
```

Part 4. Reorganization and conversion utilities

Use the reorganization and conversion utilities to reorganize databases, Fast Path DEDB areas, and HALDB partitions, and to convert databases from one type to another.

Each topic introduces one utility, describes how it works, defines the requirements and restrictions for its use, and provides examples.

© Copyright IBM Corp. 1974, 2015

Chapter 21. 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.

You can use the Database Prereorganization utility to initialize one or more HALDB partitions when converting a non-HALDB full-function database to HALDB. For initializing new or modified partitions in existing HALDB databases, the HALDB Partition Data Set Initialization utility (DFSUPNT0) is recommended; however, you can use DFSURPR0 for these purposes as well.

When reorganizing an entire HALDB or a range of partitions, you can also run DFSUPNT0 as a final step to initialize the remaining partitions that did not have any data. Partitions that receive data are dynamically initialized during reload or initial load.

When initializing partitions, the Database Prereorganization utility creates a control data set (CDS) indicating that prefix resolution and update are not required and initializes all data sets that are flagged in the RECON data set as requiring initialization.

Using the DFSURPR0 utility to initialize HALDB partitions when converting a full-function database to HALDB minimizes the amount of JCL changes required, because you can use the same JCL that you used for reorganizing the database for converting the database to HALDB.

Using the DFSURPR0 utility to initialize HALDB partitions when converting a full-function database to HALDB also avoids IMS abends caused by attempts to update partitions that have not been initialized. When a new HALDB database is first defined in the RECON data set, all of its partitions are recorded in the RECON data set as partition initialization needed (PINIT); after prereorganization the new partitions are all recorded in the RECON data set as initialized.

The following flow diagram shows the Database Prereorganization utility.

© Copyright IBM Corp. 1974, 2015

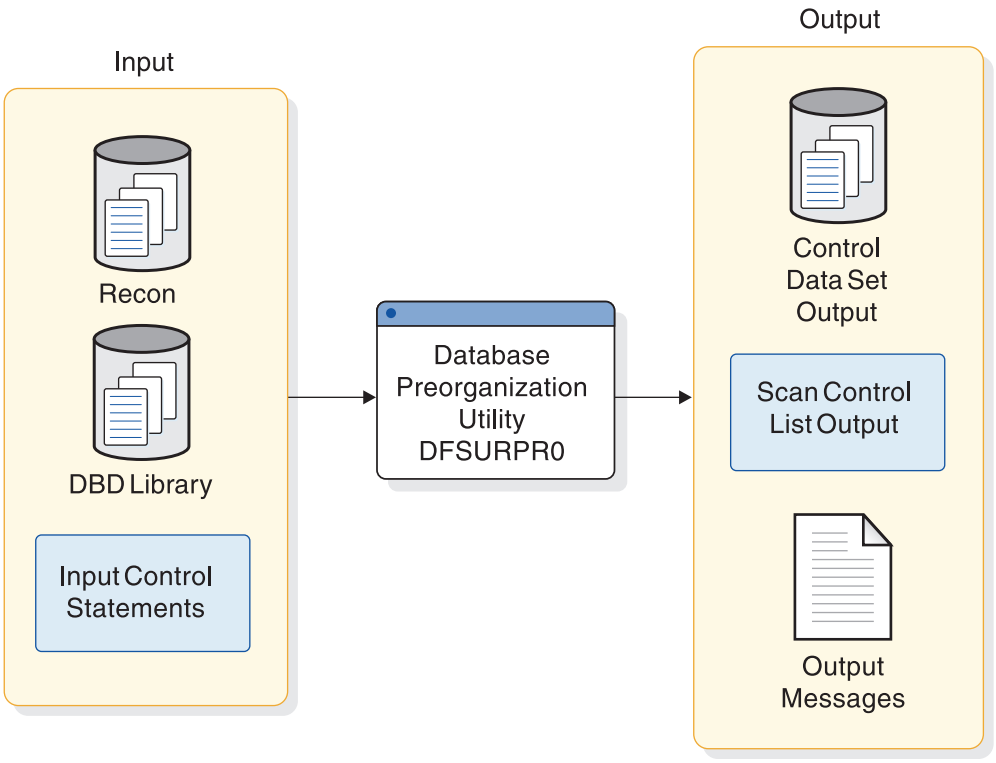


Figure 63. Database Prereorganization utility

Subsections:

- "Restrictions"
- "Prerequisites"
- "Requirements" on page 259
- "Recommendations" on page 259
- "Input and output" on page 259
- "JCL specifications" on page 260
- "Return codes" on page 262

Restrictions

The DFSURPR0 utility will work on full-function (HALDB and non-HALDB) databases only.

Utilities that alter databases cannot be run while the database is quiesced.

Prerequisites

To initialize HALDB partitions, the DFSURPR0 utility requires that they are flagged in the RECON data set as needing partition initialization. When you create or modify partition definitions in the RECON data set, the partition initialization needed flag is set automatically by IMS; however, when you delete and redefine the VSAM partition data set on DASD, you need to set the partition initialization flag in the RECON data set yourself before running the DFSURPR0 utility.

You can set the partition initialization required flag in the RECON data set for VSAM HALDB partition data sets by one of several methods:

- Use the online Partition Definition Utility (%DFSHALDB) or the DBRC online or batch commands, such as `CHANGE.DB DBD(partition_name) PINIT` and then reload the partition.
- Unload the partition, VSAM redefine it, and the reload the partition. You do not need to set the partition initialization flag in this case because PROCOPT L (load) causes the partition to be initialized internally when the first update is executed.
- Use the batch Partition Initialization utility (DFSUPNT0) to initialize the affected partitions unconditionally. The specified partition(s) are initialized regardless of their initialization state in the RECON data set.

Note: When redefining a VSAM data set for HALDB, you must include the REUSE parameter in the VSAM DEFINE statement.

Requirements

If secondary indexes exist when initially loading or reorganizing an indexed, non-HALDB 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 non-HALDB databases are being loaded or reorganized, and when both databases contain logical relationships.

If you are initializing HALDB partition data sets, you must make the RECON data sets available.

If you are initializing a VSAM data set for a newly created or changed HALDB partition, you must:

1. Ensure the 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 database.
2. Define all HALDB partition VSAM data sets with the REUSE option on the VSAM DEFINE CLUSTER statement.
3. Run the Database Prereorganization utility for an initial load, or reorganization for a reload.

Recommendations

Currently, no recommendations are documented for the DFSURPR0 utility.

Input and output

The table below identifies inputs and outputs for the Database Prereorganization utility.

Table 13. 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

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, the utility input must also include one or more non-initialized partitions. The DBDs that are

used for the databases named on the control statements must define each database as it is to exist after logical relationships are resolved. Do not modify the DBDs until after the Prefix Update utility successfully executes.

Among 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.

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.

JCL specifications

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

- A JOB statement that you define to meet the specifications of your installation.
- An EXEC statement
- DD statements that define inputs and outputs

The JCL input can also include utility control statements.

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.

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.

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.

Return codes

The following return codes are provided at program termination:

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

Related reference:

DBBBATCH procedure (System Definition)

DLIBATCH procedure (System Definition)

Control statements for the DFSURPR0 utility

You can use the following four utility control statements for the DFSURPR0 utility: DBIL, DBR, DBM, and OPTIONS.

DBIL statement

The DBIL 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.

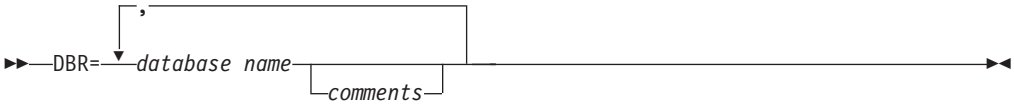
When structural changes affecting logical relationships are made to a database during a database reorganization process, the following must be performed prior to the HD reload step:

- Assemble and bind 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.

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

The DBR 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.

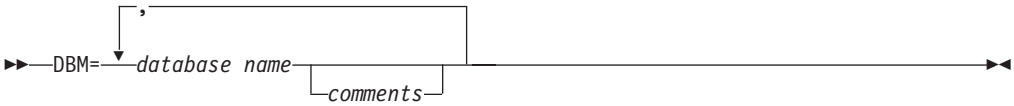
If a HISAM database is to be reorganized using the HISAM Reorganization Unload/Reload utilities, the HISAM DBD name must not be listed on a DBR control statement. If a HISAM database is to be reorganized using the HD Reorganization Unload/Reload utilities, however, the HISAM DBD name must be listed on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

The DBR utility control statement disables DB scan bit in the Control dataset (CDS) for migration of non-HALDB database to HALDB database.



OPTIONS statement

The OPTIONS 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 Database Prereorganization utility (DFSURPR0) and the Database 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.

Related concepts:

Summary on use of utilities when adding logical relationships (Database Administration)

Examples for the DFSURPR0 utility

These examples show usage for the DFSURPR0 utility.

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

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 figure below shows REUSE specified for VSAM 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
/*
```

The example in the figure below notifies DBRC that a HALDB partition requires initialization. Use the command in the example before running the Database Prereorganization utility.

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

The example in the figure below shows the job control statements required to define a VSAM 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
/*
```

Related concepts:

HALDB partition initialization (Database Administration)

Chapter 22. HALDB Migration Aid utility (DFSMAID0)

The HALDB Migration Aid utility (DFSMAID0) scans an existing full-function database and returns estimates of the number of partitions, the partitions sizes, and the partition key-range specifications that you can use when converting the scanned database to HALDB.

The DFSMAID0 utility returns different information depending on the criteria you submit to the utility:

- If you know the key ranges that you want for each partition in the planned HALDB database, you can submit the high key for each partition to the utility by using the KR= keyword in the utility control statement. The DFSMAID0 utility returns the space required for each partition.
- If you know the maximum number of bytes of segment data that you want in each partition, you can submit a byte number by using the MAX keyword in the utility control statement. The DFSMAID0 utility returns the number of partitions required in the planned HALDB database and the required high key of each partition.
- If you know how many equal-sized partitions that you want in the planned HALDB database, you can submit the number of partitions by using the NBR= keyword in the utility control statement. The DFSMAID0 utility returns the space required for each partition and the required high key of each partition.

For each type of database analysis that the DFSMAID0 utility performs (KR, MAX, or NBR), the utility traverses the IMS database only once.

The DFSMAID0 utility uses up to four z/OS data spaces of 2 GB each for storing and sorting statistics. The exact number of the data spaces that the DFSMAID0 utility requires depends on the type of analysis that the utility performs.

The DFSMAID0 utility provides a sampling option that can reduce the amount of storage and time required by the DFSMAID0 utility to process the target database. When the SAMPLE keyword is specified, the DFSMAID0 utility analyzes only a subset of the records contained in the target database and uses only three z/OS data spaces. The sampling option is not supported when you specify the KR= keyword.

Subsections:

- "Restrictions" on page 268
- "Prerequisites" on page 268
- "Requirements" on page 268
- "Recommendations" on page 268
- "Input and output" on page 268
- "JCL specifications" on page 269
- "Return codes" on page 270

© Copyright IBM Corp. 1974, 2015

Restrictions

The DFSMAID0 utility will work on full-function databases only.

Utilities that alter databases cannot be run while the database is quiesced.

Prerequisites

Currently, no prerequisites are documented for the DFSMAID0 utility.

Requirements

The DFSMAID0 utility requires the use of data spaces to store and sort the statistics generated by the analysis of the target database.

Recommendations

Use sampling, which is enabled by specifying a sample size in the SAMPLE keyword for large databases. The SAMPLE keyword can be used with the NBR or MAX EXEC statement variables. It cannot be used with the KR EXEC statement variable.

Ensure that the appropriate data spaces are defined for the DFSMAID0 utility. Data spaces are normally defined during installation. A z/OS User Step Initiation installation exit (IEFUSI) might further limit the number and storage sizes of the data spaces allocated to your job.

Input and output

The DFSMAID0 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 DFSMAID0 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

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.

The following figure is an example of the statistics report generated by the DFSMAID0 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
```

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

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

JCL specifications

The DFSMAID0 utility is executed as a standard z/OS job. The JCL specifications for the DFSMAID0 utility include a JOB statement, the EXEC statement, and the DD statements. One or more utility control statements must be/can be included with the JCL statements.

The following JCL statements are required:

- A JOB statement that you define to meet the specifications of your installation.
- 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 and PSB 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. 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. For system C programming, SYSIN becomes the C standard input (stdin) to DFSMAID0. The maximum value for LRECL is 600 bytes.

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.

Return codes

The following return codes are provided at program termination. These return codes can be found in the job step documentation at offset X'50':

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.

The DSPSERV (Create Data Space) macro return code is at offset X'00' for 4 bytes in the dump produced by message DFS2397E. The message explanation can be found in the z/OS MVS™ Programming: Authorized Assembler Services Reference. Use NNNN as part of the mask on top of the reason code xxNNNNxx to determine the source of the error.

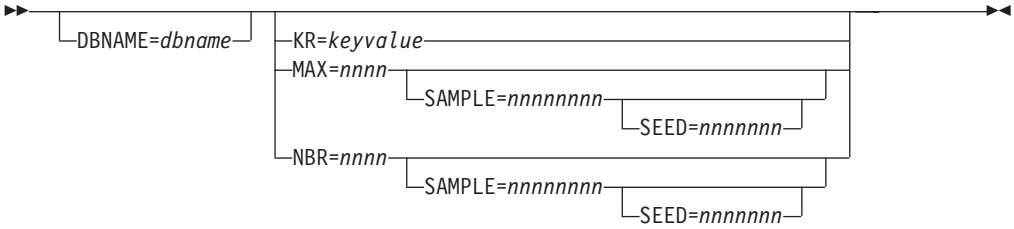
Control statement for the DFSMAID0 utility

There are six input statements you can use to describe the 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 following syntax diagram shows the format of the HALDB Migration Aid utility control statement:



DBNAME
Specifies optionally the database to be scanned.

KR KR specifies the high key values required for each HALDB partition. One KR parameter is required for each partition except the partition that contains the highest high key or the high values. The analysis always includes a high values partition. This parameter is mutually exclusive with MAX and NBR.

KR does not support the SAMPLE keyword and produces unpredictable results if both are specified.

When specifying a key value, you must specify whether the value is a hexadecimal or character value and you must enclose the key value in single quotes. For example, for a character key, code KR=C'*key_value*' and for hexadecimal key, code KR=X'*key_value*'.

The key does not need to be the same length as defined in the DBD. It will be padded with high values, for example X'FF', to fill out the complete key size.

MAX
Specifies the maximum number of bytes required for each HALDB partition in the new database. Mutually exclusive with NBR and KR.

SAMPLE
SAMPLE 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
SEED 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.

NBR
Specifies the number of HALDB partitions required in the new database. The NBR range is one to 1001. Mutually exclusive with MAX and KR.

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.

Examples for the DFSMAID0 utility

These examples show how to use the input parameters to control partitions.

Subsections:

- "Three partition definition"
- "MAXX keyword usage"
- "Sample while doing partition analysis"

Three partition definition

The input in the following example 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.

Figure 64. Defining partitions

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

MAXX keyword usage

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

Figure 65. Using MAX keyword

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

Sample while doing partition analysis

The following example 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.

Figure 66. Partition analysis

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

Data spaces and sampling for the DFSMAID0 utility

To use the DFSMAID0 utility efficiently with large IMS databases, you must understand how the DFSMAID0 utility works when specifying the different analyses.

The DFSMAID0 utility can require the allocation of up to four data spaces of 2 GB each to store and process the statistics resulting from a database analysis. Calculating the amount of data space storage used by the DFSMAID0 utility and, if possible, using a sample of the records in the target database as the basis for the analysis can help the DFSMAID0 utility use the data spaces more efficiently.

Subsections:

- "Data spaces and analysis type"
- "Calculating auxiliary storage requirements for data spaces" on page 274
- "Sampling overview and techniques" on page 274

Data spaces and analysis type

Depending on the type of analysis being performed, DFSMAID0 might store compressed statistics for every record in an IMS database and then sort the statistics in virtual memory by key. In IMS databases that can contain close to a billion records, the storing and sorting of statistics, even if the statistics are compressed, involves numerically intensive computation that can use a significant amount of CPU resources.

The use of data spaces by the DFSMAID0 utility depends on the type of analysis performed:

- When the KR keyword is specified on the input control statement, the DFSMAID0 utility does not store or sort statistics. Most of the time that the DFSMAID0 utility requires for execution is spent traversing the database by get-next calls. Sampling does not apply.
- When the MAX keyword is specified, the DFSMAID0 utility must store and sort record statistics, unless the IMS database is indexed.

  After the records statistics are in key-sequence, the DFSMAID0 utility iteratively fills consecutive partitions to the specified maximum byte count, except for the last partition.

  You can minimize the storage and time required for storing and sorting by having the DFSMAID0 utility randomly sample the entire IMS database and produce an interpolated partition analysis.
- When the NBR keyword is specified, the DFSMAID0 utility traverses the IMS database, stores compressed record statistics in data spaces, and, unless the database is indexed, sorts the record statistics by key.

  After the record count is known and the records are accessible in key-sequence, DFSMAID0 computes a maximum partition size and fills consecutive partitions to the computed maximum.

  Sampling can significantly reduce the amount of data stored when NBR is specified and can also improve sort performance dramatically.

Calculating auxiliary storage requirements for data spaces

When performing a full-database analysis, use the following calculation to determine the amount of storage needed to perform the analysis.

For *n* records in 3 or more data spaces:

```
n x ( 4 + root_keylen + 12 x no_segment_types ) + 12288
```

When using sampling analysis, use the following calculation to determine the amount of storage needed to perform the analysis.

For sample size *n* in exactly 3 data spaces:

```
n x ( 8 + root_keylen + 12 X no_segment_types ) + 12288
```

Sampling overview and techniques

The DFSMAID0 utility uses a multi-data space heap sort operation to rearrange in ascending order by key the record indices assigned by the DFSMAID0 utility. The sort computation grows in proportion to *n**log2(*n*), where *n* is the lesser of the sample size and the full record count. The heap sort operation does not block for cache, table lookaside buffer (TLB), or similar operations; sampling is more practical for the following reasons:

Because you might not be able to run a complete analysis of your database, the following examples provide a method for validating your sampling results; however, your results might vary:

- With the optional control card SAMPLE=*n*, the DFSMAID0 utility partitions a random sample of size *n* records, scaling the tabulated numbers by the ratio of the number of records in the database to the number of records in the sample. For a homogeneous database with records mostly of similar structure, sampling can be very accurate.

  For example, select 20 000 random integers from integers 1 to 2 147 483 646, sort the sample, and average the middle two: sample[10000] and sample[10001]. This experiment repeated 100 times produced estimates all within 1.78% of the actual median, 1 073 741 823.5.

  Bisecting an IMS database is a similar problem: Sort by key, and bisect at the median. A sample of size 20 000 is likely to be within 1-2% of the exact result. You do not need to store and sort orders of magnitude more data for less than 2% improvement in accuracy, especially if the database is not static. Larger databases do not require larger samples for similar accuracy.
- The scaling factor used to inflate the sample to the size of the full database is:

  ```
  (number_of_records_in_database) ÷ (number_of_records_in_sample)
  ```

  Therefore, by definition, the root segment counts in the estimated partitions equals the total number of root segments in the database, while other estimates will approximate corresponding database totals.
- You can check sampling stability by experimenting with different random *seeds*. A seed is used as a random value to prime a randomizer, for example specify "sample=10 000, seed=7". Partitioning can be checked and refined by using the key range (KR) analysis, which involves no storing or sorting of records. Again, sampling is optional but recommended for large databases.

  The randomizer output is used as an input parameter into a key-generation algorithm, which is used to access the database randomly. For best results, specify a large odd number. Prime numbers are good to use as seed values.

Chapter 23. HD Reorganization Reload utility (DFSURGL0)

The HD Reorganization Reload utility (DFSURGL0) reloads databases and HALDB partitions by using the output data sets that are created by the HD Reorganization Unload utility (DFSURGU0).

The DFSURGL0 utility supports the following types of databases:

- HDAM
- PHDAM
- HIDAM
- PHIDAM
- HISAM
- PSINDEX

The DFSURGL0 utility performs sequence checking on all applicable segments of the database records.

If a non-partitioned database includes logical relationships or secondary indexes, the DFSURGL0 utility creates a work data set. The work data set is used by the Database Prefix Resolution utility to resolve logical or secondary index relationships. You do not need to resolve logical or index relationships when reorganizing HALDB partitioned databases.

For PHDAM and PHIDAM databases, the DFSURGL0 utility provides several options for updating the indirect list data set (ILDS) in each partition. You can select an ILDS update option by specifying or omitting the DFSURGL0 utility control statements.

Your ILDS update options include:

- Updating the ILDS during the sequential load of the partition data sets each time a target segment of a logical relationship or secondary index is read by the DFSURGL0 utility. The DFSURGL0 utility updates the ILDS by using VSAM update mode. This is the default method of the DFSURGL0 utility when no utility control statement is specified.
- Updating the ILDS after the sequential load of the partition data sets is complete by using VSAM load mode and a single task. By using VSAM load mode, the DFSURGL0 utility includes in the ILDS the free space that is specified in the VSAM DEFINE statement. You can select this option by specifying the ILDSINGLE utility control statement. The DFSURGL0 utility must have five z/OS data spaces available to it for processing.
- If you are converting a non-partitioned database to HALDB, updating the ILDS after the sequential load of the partition data sets is complete by using VSAM load mode and multiple tasks. By using VSAM load mode, the DFSURGL0 utility includes in the ILDS the free space that is specified in the VSAM DEFINE statement. You can select this option by specifying the ILDSMULTI utility control statement. The DFSURGL0 utility must have five z/OS data spaces available to it for processing.

© Copyright IBM Corp. 1974, 2015

- Preventing the DFSURGL0 utility from updating the ILDS. If you select this option, you must update the ILDS by some other means, such as the HALDB Index/ILDS Rebuild utility (DFSPREC0). You can select this option by specifying the NOILDS utility control statement.

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 HALDB databases.

A flow diagram of the HD Reorganization Reload utility is shown in the following figure.

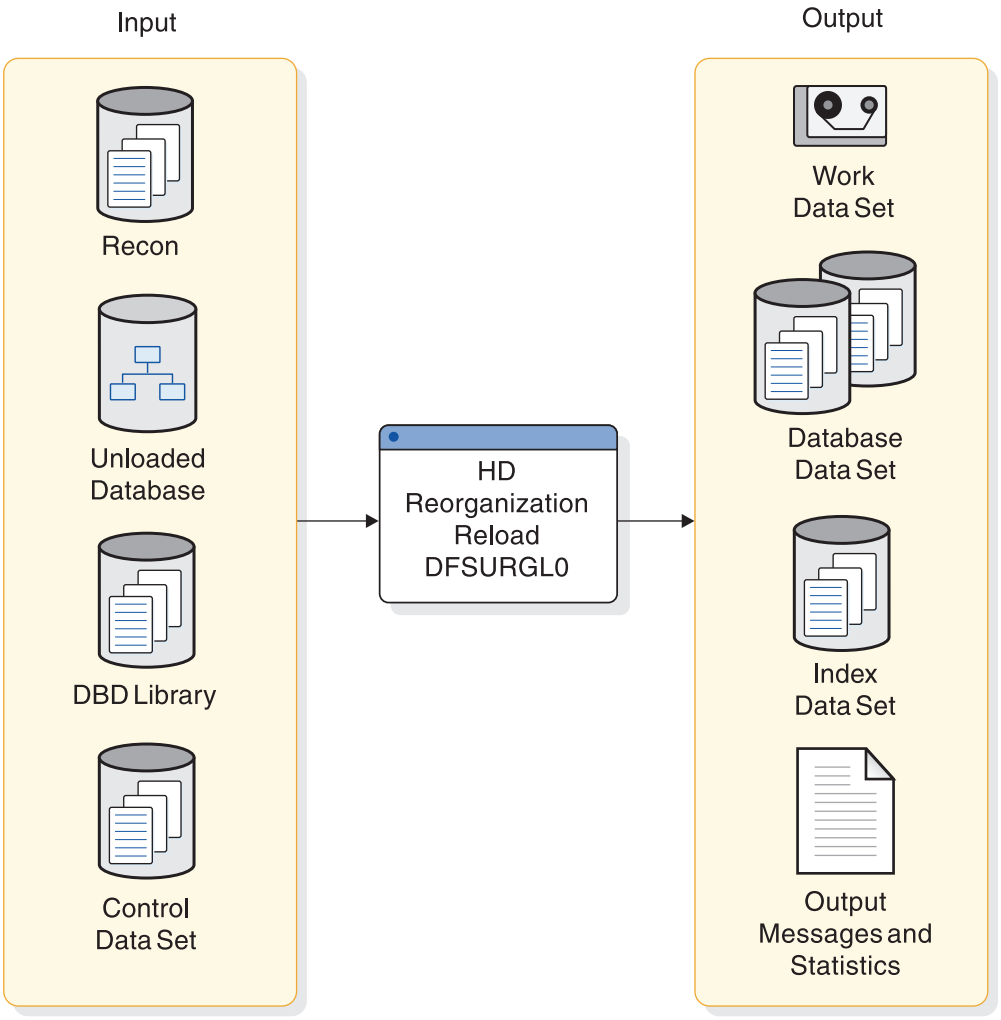


Figure 67. HD Reorganization Reload utility

Subsections:

- "Restrictions" on page 277
- "Prerequisites" on page 277
- "Requirements" on page 277
- "Recommendations" on page 278
- "Input and output" on page 280
- "JCL specifications" on page 280

• "Return codes" on page 283

Restrictions

The following restrictions apply when using the DFSURGL0 utility:

- Do not use the DFSURGL0 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 (DFSURRL0) instead.
- If you are reloading a HALDB database, you cannot use the Utility Control Facility with the DFSURGL0 utility.
- You can concatenate HALDB unload files on input to the DFSURGL0 utility, except if any of the following control statements were specified when the unload files were created by the HD Reorganization Unload utility (DFSURGU0):
  - MIGRATE
  - MIGRATX
  - FALLBACK
- The ILDSINGLE SYSIN statement can be used only for reorganizing existing HALDB database partitions. It cannot be used when converting a database to HALDB.
- The ILDSMULTI SYSIN statement can be used only for converting a database to HALDB. Do not use the ILDSMULTI SYSIN statement to reorganize an existing HALDB database partition.

The DFSURGL0 utility works on full-function (HALDB and non-HALDB) databases only.

Utilities that alter databases cannot be run while the database is quiesced.

Prerequisites

Before you can run the DFSURGL0 utility, you must create input files for the utility by running the HD Reorganization Unload utility (DFSURGU0).

If you are changing a DBD, you must complete the following tasks before you run the HD Reorganization Reload utility:

1. Assemble and bind 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.

Requirements

The DFSURGL0 has the following requirements:

- When reloading a HALDB partition with the ILDSINGLE SYSIN statement, the ILDS VSAM cluster must be deleted and redefined.

- For PHIDAM databases, you must ensure that all records are in key-sequence prior to submitting the DFSUINPT DD statements to HD Reload. HD Reload does not validate the sequence of records in input DD statements. Furthermore:
  - If you are concatenating your DFSUINPT DD statements, all records must be in key-sequence within each partition.
  - If you make changes to a partition selection exit routine, or you change to key-range partitioning after using a partition selection exit routine, you might need to re-sort all records prior to submitting the DFSUINPT DD statements to HD Reload.

Recommendations

The following recommendations apply to the use and execution of the DFSURGL0 utility:

- 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.
- When defining the DCB parameters for the DD statement for a work data set for the Prefix Resolution utility (DFSURG10), a value of LRECL=900 is recommended, but a smaller value (as small as 300) can be used if no secondary indexes are present.
- When loading a PHDAM or PHIDAM partition that has logical relationships or secondary indexes, you have several options for how the DFSURGL0 utility updates the indirect list data set (ILDS). You select your option by specifying or omitting DFSURGL0 utility control statements. The utility control statement specifications are listed with the circumstances in which their use is recommended.

  No utility control statement

  When no control statement is specified, DFSURGL0 loads the partition data sets sequentially and updates each indirect list entry (ILE) in the ILDS immediately upon loading the target segment of the ILE. DFSURGL0 updates the ILDS by using "VSAM update mode." When DFSURGL0 finishes loading the partition data sets, it is also finished rebuilding the ILDS. VSAM update mode does not add or maintain free space in the ILDS.

  Do not include a utility control statement:

  - To retain the ILEs for deleted segments for recovery purposes
  - To update an ILDS that requires only minimal updates

  ILDSINGLE

  When the ILDSINGLE control statement is specified, the DFSURGL0 utility loads partition data sets sequentially and stores the ILEs in one of five data spaces that are required by the utility for each partition. When the DFSURGL0 utility finishes loading the partition data sets, it sorts the ILEs by their indirect list keys (ILKs) and then loads all of the ILEs into the ILDS in a single pass by using VSAM load mode. Each ILE is loaded with the amount of free space that is specified in the VSAM KSDS DD statement that defines the ILDS.

The ILDSINGLE control statement requires that the DFSURGL0 utility have five z/OS data spaces available to it for processing for each partition.

Use the ILDSINGLE control statement:

- To rebuild an ILDS that requires a significant number of updates
- To eliminate CI and CA splits in the ILDS
- To add free space to the ILDS
- To remove unused ILEs that reference deleted segments
- To reduce overall execution time of subsequent reloads that run in VSAM update mode
- To improve the execution time of the DFSURGL0 utility
- For converting a database to HALDB (the ILDSMULTI option only)

When an ILDS requires heavy updating, VSAM load mode can be faster than the VSAM update mode. Also, eliminating CI and CA splits, as well as adding free space to prevent splits in the future, can improve the performance of subsequent reorganizations and recoveries of HALDB partitions.

The ILDSINGLE control statement is not supported when reloading a database as part of its conversion to HALDB.

An ILDS produced by using the ILDSINGLE option is usually larger than an ILDS produced by using the other options of the HD Reorganization utility due to the addition of free space. An ILDS produced by ILDSINGLE is equivalent to an ILDS rebuilt by the Index/ILDS Rebuild utility (DFSPREC0) when either the ILEF or the BOTHF option is specified.

ILDSMULTI

Use the ILDSMULTI control statement when you are converting a non-partitioned database to HALDB. Similar to the ILDSINGLE control statement, when the ILDSMULTI control statement is specified, DFSURGL0 loads partition data sets sequentially and stores the ILEs in one of five data spaces that are required by the utility for each partition. When DFSURGL0 finishes loading the partition data sets, it sorts the ILEs by their indirect list keys (ILKs) and then loads all of the ILEs into the ILDS in a single pass by using VSAM load mode. Each ILE is loaded with the amount of free space that is specified in the VSAM KSDS DD statement that defines the ILDS.

The ILDSMULTI control statement requires that the DFSURGL0 utility have five z/OS data spaces available to it for processing for each partition.

NOILDS

When the NOILDS control statement is specified, the DFSURGL0 utility loads the partitions without updating the ILDS. The NOILDS option provides the fastest reload of a HALDB database; however, if your HALDB database uses logical relationships or secondary indexes, you must update the ILDS by using some other means, such as the Index/ILDS Rebuild utility (DFSPREC0). You can use the NOILDS control statement for both reorganization reloads and migration reloads.

Input and output

The primary input for the DFSURGL0 utility is an unload data set created by the HD Reorganization Unload utility, although other inputs are also necessary. The primary output of the DFSURGL0 utility is a loaded database data set and can include index data sets and work data sets.

The HD Reorganization Reload utility generates output messages and statistics. The utility generates a summary report or, if you specify STAT=DET in the SYSIN data set, a detailed report.

The information in the following table identifies inputs and outputs for the HD Reorganization Reload utility.

Table 14. 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

JCL specifications

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 that define 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. The variable *dbdname* is the name of the DBD that defines the database to be reloaded. The normal IMS positional parameters such as SPIE, BUF, and DBRC can follow *dbdname*.

Related reading: For additional information on executing a batch processing region, see:

- DBBBATCH procedure (System Definition)
- DLIBATCH procedure (System Definition)

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 that contains 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, ILDSMULTI, and ILDSINGLE. The data set can reside on a tape or direct-access device, or it can be routed through the input stream. This DD statement is not necessary if no utility control statements are 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. However, you are responsible for ensuring that you include all required data sets and do not include duplicate data sets. HD Reload does not validate the data sets for you.

Requirement: For PHIDAM databases, you must ensure that all records are in key-sequence prior to submitting the DFSUINPT DD statements to the HD Reorganization Reload utility. HD Reorganization Reload utility does not validate the sequence of records in input DD statements. Furthermore:

- If you are concatenating your DFSUINPT DD statements, all records must be in key-sequence within each partition.
- If you change a partition selection exit routine, or you change to key-range partitioning after using a partition selection exit routine, you might need to re-sort all records prior to submitting the DFSUINPT DD statements to the HD Reorganization Reload utility.

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 that 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 that are 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 that is 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 that is required by the DL/I buffer handler. This statement is required.

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.

Related reading: For additional information on control statement format and buffer pool structure, see IMS buffer pools (System Definition).

DFSCTL DD
Describes the data set containing SBPARM control statements that 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.

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.

Related reading: For a description of SBPARM control statements, see Specifying sequential buffering control statements (System Definition).

SYSABEND DD or SYSUDUMP DD
Define 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.

Return codes

The following return codes are provided at program termination:

0 Database reload was completed successfully

4 There were no segments to reload

8 Reload count differs from unload count or an error encountered during ILDS processing

16 Database reload did not complete successfully

Related reference:

Chapter 30, "Utility Control Facility (DFSUCF00)," on page 369

Utility control statements for the DFSURGL0 utility

The DFSURGL0 utility accepts several control statements that you can use to specify processing options: NOILDS, ILDSMULTI, and ILDSINGLE.

The DFSURGL0 utility control statements include:

NOILDS
Disables ILDS processing during a reload of an existing or newly converted HALDB database.

Using NOILDS avoids I/O to the ILDS data set whenever a new ILE is required. When this option is specified, the ILDS associated with each reloaded partition is recorded in the RECON data set as needing to be recovered. Execute the ILDS Rebuild utility (DFSPREC0) to rebuild the ILDS in a separate job. This may lengthen overall execution time as the entire HALDB partition is read in order to build each ILDS. Use this statement for large unload files and run DFSPREC0 in separate job for each partition to benefit from multitasking. The NOILDS technique is valuable for migration unload files.

ILDSMULTI
Enables multi-task ILDS processing for migration of a Full Function DL/I data base to a HALDB database.

Using ILDSMULTI causes multiple TCBs to be attached to sort the ILE records in ILK sequence. The DFSURGL0 utility requires a total of five data spaces for each partition being loaded. The data spaces are extended as required. Some system tuning of data spaces may be required when using this option. ILDSMULTI is used only for migration reloads.

ILDSINGLE
Enables a single task to sort ILEs (Indirect List Entries) in ILK (Indirect List Key) sequence for a specific partition. Using this option includes free space in the ILDS and shortens reload execution time.

Using ILDSINGLE causes the ILEs to be passed to a single TCB for sorting. The sorting TCB requires a total of five data spaces for each partition. The data spaces are extended as required. Some system tuning of data spaces might be required when using this option.

ILDSINGLE execution requires deleting and redefining VSAM cluster prior to execution. KSDS free space is honored, avoiding CA or CI splits of the ILDS during reload. The ILEs are sorted in ILK sequence and inserted into the ILDS

after the reload of its associated partition completes. If an error is encountered during processing an identifying message is printed and the ILDS is recorded in the RECON data set as recovery needed. Use the ILDS rebuild utility to rebuild the ILDS after encountering an error.

If ILDSINGLE=*partname* is specified and the ILDS has data, then after reload of the target partition completes successfully, error message DFS871E is issued when the first duplicate key is entered.

For example:

```
DFS871E  VSAM ERROR ENCOUNTERED LOADING ILDS WITH
DDNAME=partname RC= 00000008 REASON= 00000008
```

The ILDS is then recorded in the RECON data set as recovery needed. Any subsequent attempt produces U3303 abend before accessing the HALDB partition associated with the ILDS. Messages DFS047A, DFS3303I and DFS0832I provide useful information about the error. To reload the partition and associated ILDS you must first use the DBRC command CHANGE.DBDS DBD(partname) DDN(ILDSddn) NORECOV and then delete and redefine the ILDS VSAM cluster.

ILDSINGLE cannot be used for migration unload files. Use ILDSMULTI instead.

Unloaded data for multiple HALDB partitions in a single file may be used as input. Multiple ILDSINGLE jobs may execute in parallel if the unload data set exists on DASD. Processing is isolated to the target partition specified with ILDSINGLE=partname, other partitions are ignored during processing. The unload report will reconcile the unloaded and the reloaded data. No messages and no keys are printed unless an FM status code is returned for the target partition.

Output Messages and Statistics for DFSURGL0

The HD Reorganization Reload utility provides output messages and statistics.

The statistics appear in the output after the messages and under the heading SEGMENT LEVEL STATISTICS appears.

The fields in the statistics report include:

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. Segments inserted by a user exit are not included in this total.

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. Segments inserted by a user exit are not included in this total.

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

The following figure 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
```

If you are loading a HALDB database and request detailed reports by specifying STAT=DET in the SYSIN control statement, the report includes a list of the segments loaded for each partition and then a listing of the total segments loaded for the entire HALDB 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.

The following figure is an example of single partition statistics reconciled against an input unload file containing unload data from other partitions.

```
ILDSINGLE=PDHDOJA

                SEGMENT  LEVEL  STATISTICS

                            TOTAL SEGMENTS BY SEGMENT TYPE
 SEGMENT      SEGMENT
  NAME         LEVEL            RELOADED     DIFFERENCE
```

```
J1              1               72              32 -

J2              2                9              64 -

J3              3                3              32 -

J4              4                3              32 -

J5              2               16              32 -

J6              3               12              32 -

J7              4                6              32 -

J8              5                3              32 -

J9              2               16              64 -

J10             3                9

J11             4                0
J7P             3                6              32 -

J12             2               79              64 -

J13             3               66               2 -

J14             4               40

J13X            3               13              32 -

J15             3               41


          TOTAL  SEGMENTS  IN  DATA  BASE
       UNLOADED          RELOADED      DIFFERENCE

            876              394             482
```

Examples for DFSURGL0

These examples show sample JCL for the DFSURGL0 utility.

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

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

Subsections:

- "JCL for an HDAM reorganization reload"
- "HIDAM reorganization reload JCL" on page 287
- "HIDAM VSAM database unload and reload" on page 287
- "HIDAM HALDB migration reload with NOILDS" on page 288
- "HIDAM HALDB migration reload with ILDSMULTI" on page 289

JCL for an HDAM reorganization reload

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
```

HIDAM reorganization reload JCL

This example shows the JCL for a HIDAM reorganization reload. The primary index database data sets are also described. The RECON data sets are not used.

```
//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
```

HIDAM VSAM database unload and reload

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
/*
```

HIDAM HALDB migration reload with NOILDS

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
```

HIDAM HALDB migration reload with ILDSMULTI

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 24. HD Reorganization Unload utility (DFSURGU0)

Use the HD Reorganization Unload utility (DFSURGU0) to reorganize IMS full-function databases, to change IMS full-function database structures, and to record information about logical relationships that are used by IMS full-function databases.

The DFSURGU0 utility is usually paired with the HD Reorganization Reload utility (DFSURGL0), which reloads IMS full-function databases by reading the unload data set that is created by the DFSURGU0 utility.

You can use the DFSURGU0 utility for the following purposes:

- Unload an HDAM, PHDAM, HIDAM, PHIDAM, PSINDEX, or HISAM database to a sequential data set.
- Unload a subset of PHDAM, PHIDAM, or PSINDEX partitions by including the appropriate utility control statement. If you do not include a utility control statement when running this utility against a HALDB database, the DFSURGU0 utility unloads the entire database.
- 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 that are associated with logical parents in a database.
- Convert databases to HALDB partitioned databases and fall back from HALDB.

The DFSURGU0 utility can also make certain structural changes to a database during the process of reorganization if you replace the original DBD of the database that is being reorganized with a new version of the DBD that reflects the structural changes.

The structural changes that you can make with the DFSURGU0 utility include:

- Delete an existing segment type from the DBD, provided all segments of this type were deleted from the database prior to execution of the HD Reorganization Unload utility.
- Add new segment types 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.
- Change, add, or delete any field statement except the one for the sequence field of a segment; however, IMS makes no attempt to alter the data content of a segment.
- Change existing segment lengths 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 your responsibility to replace the extended portion of the segment by running an application program in update mode under IMS.
- Change the DL/I access method. OSAM format can be changed to VSAM format or VSAM format to OSAM. Any DL/I access method can be changed except for HDAM or PHDAM, which cannot be changed to either indexed method. HISAM, HIDAM, and PHIDAM can be changed to HDAM or PHDAM.

© Copyright IBM Corp. 1974, 2015

- Change segment pointer options for HDAM, PHDAM, HIDAM, and PHIDAM databases. 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 databases.

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.

When unloading databases, the HD Reorganization Unload utility adds a prefix to each unloaded segment to support reorganization. The prefix that is created for segments unloaded from a non-HALDB database is different that the prefix created for segments that are unloaded from HALDB databases.

The prefix that is created for segments that are unloaded from non-HALDB databases is mapped by the macro DFSURGUF.

The prefix that is created for segments that are unloaded from a HALDB database is mapped by the macro DFSURGUP. DFSURGUP includes the indirect list key (ILK) and the extended pointer set (EPS) if the unloaded segment is a logical child.

Usually, an ILK contains the relative byte address (RBA), the partition ID, and the partition reorganization number of the segment when it was first created, as shown in the following figure. However, if you are migrating an HD database to HALDB, the DFSURGU0 utility creates a migration ILK that contains a partition ID of zero and the field that normally contains the reorganization number contains the number of the data control block (DCB) on DASD from which the target segment was read.

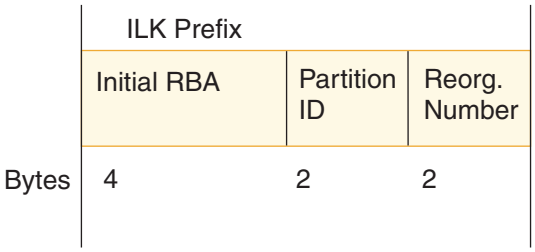


Figure 68. Format of an ILK

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

The following figure is a flow diagram of the HD Reorganization Unload utility.

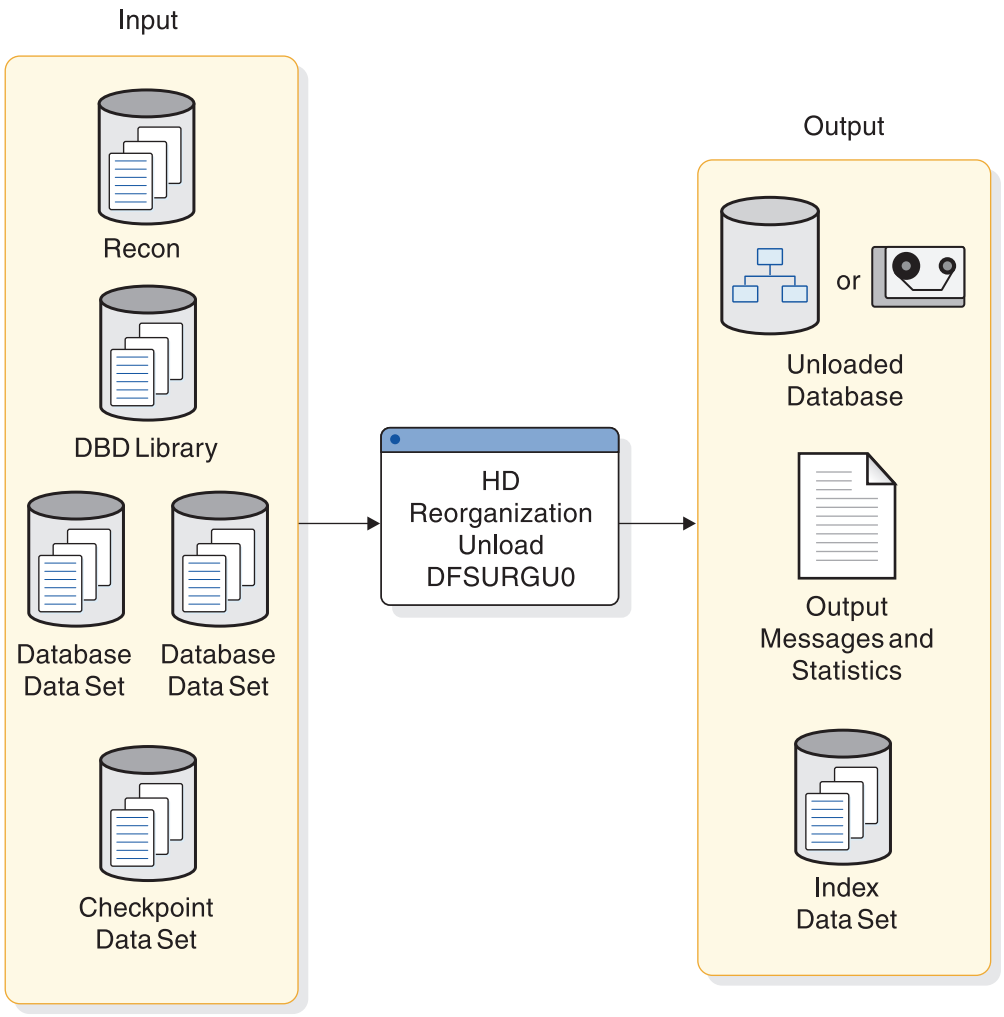


Figure 69. HD Reorganization Unload utility

Subsections:

- "Restrictions"
- "Prerequisites" on page 294
- "Requirements" on page 294
- "Recommendations" on page 296
- "Input and output" on page 296
- "JCL specifications" on page 298
- "Return codes" on page 300

Restrictions

The following restrictions apply to the DFSURGU0 utility:

- The Utility Control Facility is not supported for HALDBs.
- After running the DFSURGU0 utility, do not update the database until the database is reloaded. Any updates that are made after the unload files are created will be lost if the unload files are used to reload the database.
- Names of existing segment types must not be changed.
- An unload that uses the MIGRATE control statement can be performed only on non-HALDBs.

- You cannot change the DL/I access method of a HDAM or a PHDAM database to use an indexed access method.
- 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.
- The DFSURGU0 utility will work on full-function (HALDB and non-HALDB) databases only.
- Utilities that alter databases cannot be run while the database is quiesced.

The following restrictions apply to the DFSURGL0 utility when the KEYRANGE option is used:

- VSAM data set sharing options must allow multiple readers.
- To prevent unpredictable results, key ranges must start and end on partition boundaries. One or more partition's worth of data can be unloaded, but must be unloaded into one data set as HD Reload supports only a single data set as input.
- The KEYRANGE option is only valid after a MIGRATE= statement. It is not valid after a MIGRATX= statement.
- The KEYRANGE option requires unload of the DLI database with IRLM=N, DBRC=N and DISP=SHR for database data sets.
- The migration reload must use DBRC=Y and requires allocating the RECON data sets containing the new HALDB partition definitions, either with JCL or dynamically using DBRC RECON MDA members.

Prerequisites

The prerequisites for the DFSURGU0 utility include:

- If you are deleting an existing segment type from a DBD, before you execute the DFSURGU0 utility, you must first delete all segments of the type to be deleted from the database.

Requirements

The DFSURGU0 requirements include:

- The DFSURGU0 utility must be executed against the DBD describing the current structure of the database
- 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.

- To change the data set format of a HALDB database from OSAM format to VSAM format or VSAM format to OSAM format, you must delete and redefine the HALDB definition in the RECON data set.
- When unloading an HDAM or PHDAM database for a MIGRATX HD unload, the randomizing module and any IMS user exit routines, such as compression and sparse index, must be included in the JOBLIB. A sparse index is not required for a normal HD unload.
- If you are increasing the length of a fixed length segment length, you must replace the binary zeros that are used as filler in the extended portion of the segment by using an application program running in update mode under IMS.
- 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: Because virtual pairing is not supported by HALDB, the requirement to delete all segments that will become virtual does not apply to HALDB.
- If you are converting a secondary index database that has non-unique keys to HALDB, you must include separate JCL steps to sort and merge the unload records and create new /SX values prior to inputting them into the HD Reorganization Reload utility. 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.
- If you are converting a secondary index database that uses symbolic pointing to HALDB, you must include separate JCL steps to sort and merge the unload records. For examples of the sort and merge JCL, see "Sort and merge unload records for secondary indexes that use symbolic pointing" on page 305.
- If you are converting a database that has secondary indexes to a HALDB partitioned database, you must run the DFSURGU0 utility to unload each secondary index database. The DFSURGU0 utility reads the primary database to resolve the source and target segments that are needed to construct the indirect list entry key (ILK) and extended pointer set (EPS) information for the conversion unload record that is destined for the new PSINDEX. The 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.
- When you use the HD Unload utility to migrate from non-HALDB's to HALDB's, you can increase performance of MIGRATE= by using the KEYRANGE option to run multiple unload jobs in parallel. This is most useful when migrating non-HALDB's that are very large, or have a large amount of logically related segments.
- For the KEYRANGE option:

- Both fromkeyval and tokeyval should be copied from output of a DBRC LIST.DB batch command and pasted into the HDUNLOAD SYSIN DD statement.
- VSAM data set share options of (3,3) are recommended.
- To maximize performance, unload a key range of a single partition in one step followed by a migration reload of the corresponding HALDB partition.

•

Recommendations

Currently, there are no recommendations documented for the DFSURGU0 utility.

Input and output

The primary input for the DFSURGU0 utility is a database data set, although other inputs are also necessary. The primary output of the DFSURGU0 utility is an unload data set. Other output can include a checkpoint data set and output messages and unload statistics.

The information in the following table identifies inputs and outputs for the HD Reorganization Unload utility.

Table 15. 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	

The HD Reorganization Unload utility provides output messages and statistics. An example of the messages and statistics obtained from this utility is shown in the following figure.

Figure 70. Example of output messages and statistics—HD Reorganization Unload utility

```
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
```

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 that are generated as a result of the options that are selected for this execution.

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.

JCL specifications

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 to meet the specifications of your installation.
- An EXEC statement
- DD statements defining inputs and outputs

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

EXEC statement

The EXEC statement must be in the form:

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

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*.

DD statements

The input, output, and resource data sets used by the DFSURGU0 utility are identified by DD statements.

The DFSURGU0 utility uses the following required and optional 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 that contains 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
Optional data set that defines the input control statement data set only to

HALDB and fallback. If the SYSIN data set is omitted, the entire HALDB is unloaded. If the SYSIN data set is specified but does not contain anything, the entire HALDB is unloaded. 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 you are unloading a HALDB database, do not include the database DD statement. HALDB uses dynamic allocation. If you use the integrated HALDB Online Reorganization function, dynamic allocation automatically selects the appropriate A–J or M–V prefix for the data set name.

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.

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.

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 are 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.

Return codes

The DFSURGU0 utility issues one or more of the following return codes upon completion:

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.

Related reference:

Chapter 30, "Utility Control Facility (DFSUCF00)," on page 369

Chapter 27, "HISAM Reorganization Unload utility (DFSURUL0)," on page 327

DBBBATCH procedure (System Definition)

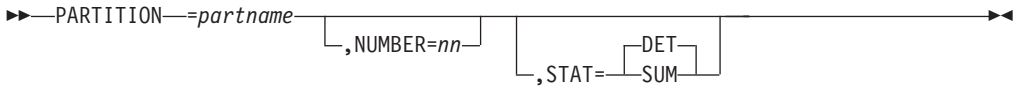
DLIBATCH procedure (System Definition)

Control statements for the DFSURGU0 utility

You can use the following utility control statements for the DFSURGU0 utility: PARTITION, MIGRATE, KEYRANGE, MIGRATX,and FALLBACK

All of the utility control statements that are used by the HD Reorganization Unload utility (DFSURGU0) specify options for processing HALDB databases or for converting databases to HALDB.

The PARTITION utility control statement must conform to the following:



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.

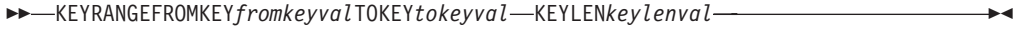
STAT=
Specifies whether to generate partition statistics. Partition statistics are generated if STAT=DET. Partition statistics are not generated if STAT=SUM. The default setting is STAT=DET.

SUM produces a summary report for all partitions processed. This option suppresses detail reports.

DET produces a detail report. The detail report includes the statistics by partition in addition to the summary report. This is the default.

This utility control statement is provided in a SYSIN stream.

MIGRATE=
Specifies whether to unload a non-HALDB database for migration. A DD card is required for all of the related secondary index databases.



KEYRANGE
KEYRANGE is designed to increase the performance of Unload, when migrating a non-HALDB to a HALDB, by running multiple unload jobs in

parallel. Invoke this function by adding a KEYRANGE statement in column 1 immediately after a MIGRATE= statement. KEYRANGE is not valid after a MIGRATX= statement.

Specify key ranges using the FROMKEY, TOKEY, and KEYLEN parms.

fromkeyval
Use either zeros, or the high key of a previous partition definition, expressed in hexadecimal.

tokeyval
The high key of the destination partition's definition expressed in hexadecimal.

keylenval
The length of the key specified as an alphanumeric value.

MIGRATX=
MIGRATX creates unload files for secondary index datasets in addition to the primary data set. It eliminates the need to run multiple migration HD Reorganization Unload utility 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 the HD Reorganization Reload utility to load the PSINDEX. DFSWRKnn and DFSSRTnn DD cards are required for each secondary index database referencing the primary database being unloaded.

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

Examples for the DFSURGU0 utility

These examples show usage for the DFSURGU0 utility.

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

Subsections:

- "Unload two HALDB partitions" on page 303
- "Reorganize a HIDAM database using the checkpoint facility and two output copies" on page 303
- "Use a checkpoint data set to restart after an abnormal termination" on page 304
- "Unload a database and two secondary indexes" on page 304
- "Sort and merge unload records for secondary indexes that use symbolic pointing" on page 305
- "Specify Small Keys" on page 306
- "Migrate a range of data using the new partition definitions:" on page 307
- "Migration for long keys (up to 256 bytes)" on page 307

Unload two HALDB partitions

In this example, partition PDHDOJB will be unloaded, followed by the next partition in partition selection order. RECONs will be dynamically allocated using MDA members from STEPLIB.

```
//HDUNLDCS JOB 'UNLOAD',MSGCLASS=A,MSGLEVEL=(1,1),CLASS=K,
//            REGION=2M,TIME=1440
//JOBCAT   DD DSN=VCATSHR,DISP=SHR
//HDUNLOAD EXEC PGM=DFSRRC00,REGION=2048K,
//            PARM=(ULU,DFSURGU0,DBHDOJ01,,,,,,,,,,)
//STEPLIB  DD DSN=IMSTESTL.TNUC0,DISP=SHR
//         DD DSN=IMSBLD.RESLIB,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//DFSRESLB DD DSN=IMSBLD.CRESLIB,DISP=SHR
//DFSURGU1 DD DSN=HALDB2-3.UNLOAD,DISP=(,CATLG),
//            UNIT=SYSDA,SPACE=(CYL,(5,3)),DCB=BUFNO=5,VOL=SER=000000
//IMS      DD DSN=IMSTESTS.DBDLIB,DISP=SHR
//         DD DSN=IMSTESTG.DBDLIB,DISP=SHR
//SYSIN    DD *
PARTITION=PDHDOJB,NUMBER=2
/*
//DFSVSAMP DD *
VSRBF=1024,40
VSRBF=4096,40
VSRBF=8192,80
/*
```

Recommendation: IBM does not recommend coding DD names for HALDB partitions because they are dynamically allocated.

Reorganize a HIDAM database using the checkpoint facility and two output copies

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.

Use a checkpoint data set to restart after an abnormal termination

In this example, execution of Example 1 is restarted after an abnormal termination. The checkpoint data set is 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.

Unload a database and two secondary indexes

The HD Reorganization Unload utility 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 the figure below 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
```

The JCL in the figure below 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
```

Sort and merge unload records for secondary indexes that use symbolic pointing

Follow these steps for writing code to sort the HD Reorganization Unload utility records when converting a secondary index databases with symbolic pointing to HALDB:

1. Get the offset to the key and the key length of the concatenated key from the 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 sort key 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 sort key 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'0290'),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
```

Specify Small Keys

In this example, DLI data is unloaded in key order by retrieving the segments greater than 00000 and less than or equal to K2000 (key > lowkey & key <= highkey) and inserted sequentially using the RECON definitions that produced the RECON listing. These examples are for a HALDB with no user partition selection exit.

1. LIST.DB of the HALDB partition definition
2. RECON listing for the first partition

```
PARTITION HIGH KEY/STRING (CHAR):  (LENGTH=5  )  K2000
PARTITION HIGH KEY/STRING (HEX):  D2F2F0F0F0
```

3. Specify a SYSIN statement for HDUNLOAD using the hexadecimal notation of the key values taken from the RECON listing. The KEYRANGE statement starts in column 1 and may be continued using "-" as a continuation character for key words but not key values contained within parentheses and may not appear beyond column 71.

```
MIGRATE=YES
KEYRANGE FROMKEY(F0F0F0F0F0) TOKEY(D2F2F0F0F0) KEYLEN(5)
```

or

```
KEYRANGE FROMKEY -
(F0F0F0F0F0) -
TOKEY -
(D2F2F0F0F0) -
KEYLEN(5)
```

4. Input the unload file to HDRELOAD.

Migrate a range of data using the new partition definitions:

In this example, DLI data is unloaded in key order by retrieving the segments greater than K2000 and less than or equal to K6000 (key > lowkey & key <= highkey) and inserted sequentially using the RECON definitions that produced the RECON listing.

1. LIST.DB of a range of HALDB partition definitions (2 to 6)
2. RECON listing for first through fourth partitions

```
PARTITION HIGH KEY/STRING (CHAR):              (LENGTH=5  )
  K2000
PARTITION HIGH KEY/STRING (HEX):
  D2F2F0F0F0

PARTITION HIGH KEY/STRING (CHAR):              (LENGTH=5  )
  K6000
PARTITION HIGH KEY/STRING (HEX):
  D2F6F0F0F0
```

3. Specification of SYSIN statements to unload using the hexadecimal notation of the key values as defined in RECON.

```
|...+....1....+....2....+....3....+....4....+....5....+....6..
MIGRATE=YES
KEYRANGE FROMKEY(D2F2F0F0F0) TOKEY(D2F6F0F0F0) KEYLEN(5)
```

4. Input the unload file to HDRELOAD.

Migration for long keys (up to 256 bytes)

Include the following to code migration for long keys (up to 256 bytes).

1. RECON listing.

```
PARTITION HIGH KEY/STRING (CHAR):              (LENGTH=129)
  J2000..........2.............3.............4...........
  ....5.............6.............7.............8.......
  .........9

PARTITION HIGH KEY/STRING (HEX):
  D1F2F0F0F0000000000000000000000F2000000000000000000000000000
  0000F3000000000000000000000000000F4000000000000000000000
  00000000F5000000000000000000000000000F6000000000000000000
  000000000000F7000000000000000000000000000F8000000000000000
  0000000000000000F9
```

2. Migration unload SYSIN key values for longer keys begin on the line following the KEYRANGE key word value in column 1 and wrap after column 65.

```
|...+....1....+....2....+....3....+....4....+....5....+....6.
MIGRATE=YES
KEYRANGE FROMKEY (-
F0F0F0
```

Chapter 25. 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 DBFUHDR0 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. The use of real storage gives improved performance and reduced logging.

Each UOW being reorganized is held exclusively by the DBFUHDR0 utility until it is reorganized and written back to its permanent location. The UOW 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 DBFUHDR0 utility control statements are submitted in the form of DEDB online utility commands.

Subsections:

- "Restrictions"
- "Prerequisites"
- "Requirements" on page 310
- "Recommendations" on page 310
- "Input and output" on page 310
- "JCL specifications" on page 311
- "Return codes" on page 312

Restrictions

You must execute the High-Speed DEDB Direct Reorganization utility in a separate job step. The program cannot switch from one utility or one database to another within the same job step.

The DBFUHDR0 utility will work on Fast Path (DEDB) databases only.

Utilities that alter databases cannot be run while the database is quiesced.

Prerequisites

Currently, no prerequisites are documented for the DBFUHDR0 utility.

© Copyright IBM Corp. 1974, 2015

Requirements

The optional data set used for segment shunting must be fixed block with LRECL=BLOCKSIZE=80.

Recommendations

Currently, no recommendations are documented for the DBFUHDR0 utility.

Input and output

The primary input to the DBFUHDR0 utility is an area data set that needs to be reorganized. The primary output of the DBFUHDR0 utility is a reorganized area data set.

The following table shows the inputs to and the outputs from the DBFUHDR0 utility:

Table 16. Input to and output from the DBFUHDR0 utility

Input	Output
An area data set that needs to be reorganized	An area that contains logically sequenced UOWs with no storage fragmentation
A DEDB utility command data set	Output messages and statistics

The High-Speed DEDB Direct Reorganization utility uses a data set that contains input parameters supplied by DEDB commands as input.

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

Example output report

The following example shows the DBFUHDR0 utility output report:

```
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
```

JCL specifications

The DBFUHDR0 utility is executed as a standard z/OS job. The JCL specifications for the DBFUHDR0 utility include a JOB statement, the EXEC statement, and the DD statements. One or more utility control statements must be/can be included with the JCL statements.

EXEC statement

The EXEC statement for the DBFUHDR0 utility can either invoke the FPUTIL procedure, which contains the required JCL, or be specified in the following format:

```
PGM=DFSRRC00
```

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.

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.

Example JCL

The following figure 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
```

Return codes

The DBFUHDR0 utility generates one or more of the following return codes:

Code	Meaning
0	Utility executed as requested.
2	Error in the INDD detected during segment shunting. Utility executes successfully; see utility output.
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.

Control statements for the DBFUHDR0 utility

The DBFUHDR0 utility control statements are submitted in the form of DEDB online utility commands. The commands are coded in the SYSIN DD statement. The control commands are also used to describe the input area data set that is being reorganized.

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.

The following commands are applicable to the DBFUHDR0 utility:

AREA **(required)**
: 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 - 8 characters. The first character must be alphabetic, and the rest of the characters are alphanumeric. Alphabetic characters include @, #, and $.

BUFNO (optional)
For the DBFUHDR0 utility, the DEDB utility command BUFNO specifies a number of *buffer sets*, rather than a number of buffers. 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.

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 "Buffers and the BUFNO command" on page 315 for information on how BUFNO is used for the High-Speed DEDB Direct Reorganization utility.

ERRORACTION **(optional)**
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.

GO **(optional)**
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.

STARTUOW **(optional)**
Specifies the first *unit of work* (UOW) to reorganize. This command is optional and is used with the reorganization utility.

STOPAFTERFAIL# **(optional)**
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.

STOPUOW
Specifies the *last unit of work* (UOW) to reorganize. This command is optional and is used with the reorganization utility.

TYPE **(required)**
Specifies either the scan, delete, or reorganization utility as the type of run.

Attention: 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 keyword for the TYPE command used to invoke the DBFUHDR0 utility is REORG, along with the synonyms HSR, HSREORG, DR and R.

You can reorganize a part of the root addressable portion of the area by specifying the UOW at which to start reorganizing by using the STARTUOW command. 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 command is not specified, the utility starts reorganizing with the first UOW in the area and continues to the specified STOPUOW command. If STOPUOW command 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.

Running the DBFUHDR0 utility

Operational considerations of the DBFUHDR0 utility include BUFNO command, segment shunting, recovery and restart, and error processing.

Subsections:

- "Buffers and the BUFNO command"
- "Segment shunting" on page 316
- "Recovery and restart" on page 317
- "Error processing" on page 318

Buffers and the BUFNO command

For the DBFUHDR0 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.

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 DBFUHDR0 utility dynamically extends the private buffer pool to obtain more buffer sets if waits for buffers are experienced. The number of extensions that are

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 that are 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 determines 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 DBFUHDR0 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 that is used can be significant. For example, an area with UOWs altering 45 CIs, each CI being 16 KB in size, using the default BUFNO specification, would require 2160 KB 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.

Segment shunting

Segment shunting is the capability to reorganize specified segments directly into dependent overflow (DOVF) or independent overflow (IOVF), bypassing copying these segments into the root anchor point control intervals (RAP CI) even though space might currently exist in the RAP CI. Segment shunting 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 the input data set INDD. This data set must be a fixed block with a 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 is 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, ERRACTN=EXIT, and ERRACTN=TERM.

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.

When ERRACTN=TERM is specified, if any cards in the INDD data set contain invalid data, the entire data set is ignored. The reorganization job will not be executed and DL/I status Z0 is returned.

AREA=areaname
Starts the input stream for a single area. 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 if 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.

Recovery and restart

If the DBFUHDR0 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. However, 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 DBFUHDR0 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 DBFUHDR0 utility, no difference in processing exists between REST=00 and REST=01.

Error processing

The following table summarizes the I/O error handling by the DBFUHDR0 utility and the area status after an I/O error.

Table 17. Summary of I/O error handling by the DBFUHDR0 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

Related reference:

FPUTIL procedure (System Definition)

Chapter 26. HISAM Reorganization Reload utility (DFSURRL0)

Use the HISAM Reorganization Reload utility to reload a HISAM database by using the output data sets of the HISAM Reorganization Unload utility (DFSURUL0)

You can also use the DFSURRL0 utility for the following additional purposes:

- 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 HISAM Reorganization Reload utility generates a statistics report on every execution. You can suppress the statistics report by using the NSTAT parameter on the OPTIONS utility control statement.

The following figure shows a flow diagram of the HISAM Reorganization Reload utility.

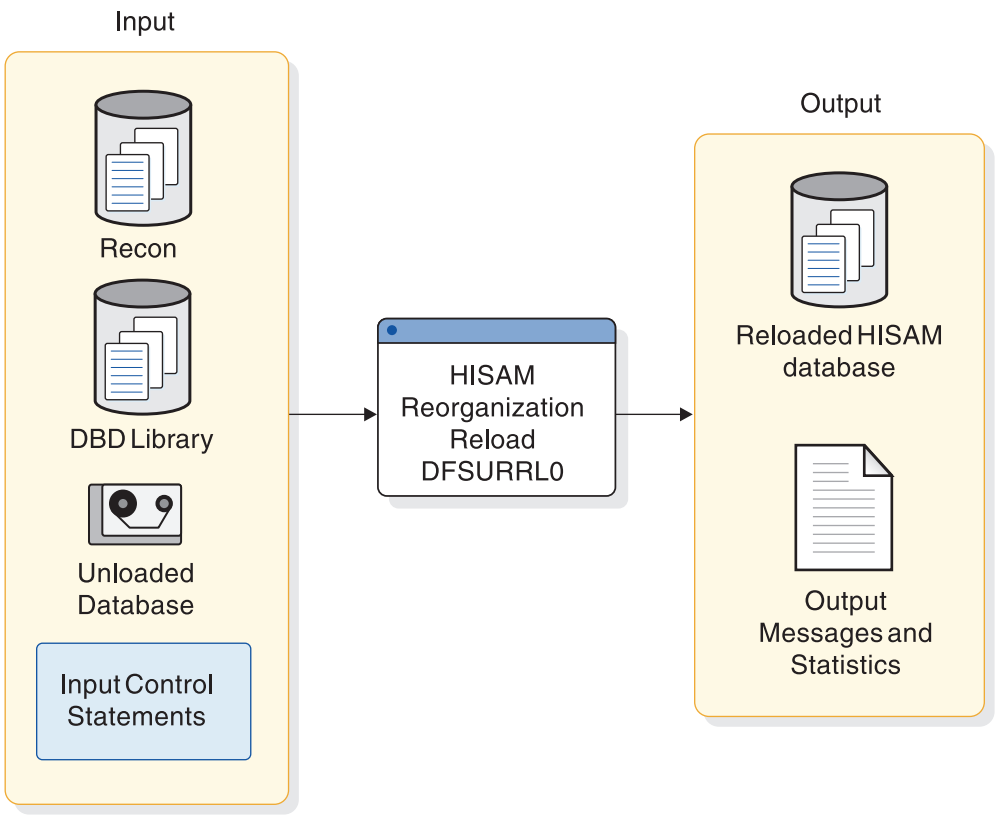


Figure 71. HISAM Reorganization Reload utility (DFSURRL0)

Subsections:

- "Restrictions" on page 320
- "Prerequisites" on page 320
- "Requirements" on page 320

© Copyright IBM Corp. 1974, 2015

- "Recommendations"
- "Input and output"
- "JCL specifications" on page 323
- "Return codes" on page 324

Restrictions

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

- You cannot use this utility to reload a PSINDEX database.
- You can use this utility only to make changes to logical record length and block size. Use the HD Reorganization Reload utility (DFSURGL0) 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 (DFSURGU0) and HD Reorganization Reload (DFSURGL0) utilities must be used instead.
- The DFSURRL0 utility works on full-function non-HALDB databases only.
- Utilities that alter databases cannot be run while the database is quiesced.

Prerequisites

Currently, no prerequisites are documented for the DFSURRL0 utility.

Requirements

You must use the same IMS release to reload the database as was used to unload the database.

Recommendations

Currently, no recommendations are documented for the DFSURRL0 utility.

Input and output

The primary input to the DFSURRL0 utility is an unload data set created by the HISAM Reorganization Unload utility (DFSURUL0). The primary output of the DFSURRL0 utility is a reloaded data set.

The following table identifies the inputs and outputs for the HISAM Reorganization Reload utility.

Table 18. Data sets used by the DFSURRL0 utility

Input	Output
RECON	VSAM KSDS data set for reloaded HISAM database
Unloaded database	Possibly an overflow OSAM data set
DBD library	Output messages and statistics
Input control statements	

Example output report

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 the following figure.

```
          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
```

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

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 NSTAT parameter on the OPTIONS utility control statement.

Following the messages for each data set group, the heading DATA SET GROUP STATISTICS appears.

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 SEGMENT LEVEL STATISTICS section contains the following fields.

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.

The TOTAL SEGMENTS IN DATA SET GROUP section contains the following fields.

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

JCL specifications

The DFSURRL0 utility is run as a standard z/OS job. The JCL specifications for the DFSURRL0 utility include a JOB statement, the EXEC statement, and the DD statements. An optional utility control statement can be included with the JCL statements.

The following JCL statements are required:

- A JOB statement that you define to meet the specifications of your installation.
- 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.

DD statements

The DFSURRL0 utility uses a number of required and optional DD statements.

The DFSURRL0 utility DD statements include:

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: 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.

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.

Return codes

The DFSURRL0 utility generates one or more of the following return codes:

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

Code	Meaning
0	All operations have successfully completed.
4	One or more warning messages were issued.
8	One or more operations did not complete successfully.

16 A severe error causing program termination occurred.

Related concepts:

IMS buffer pools (System Definition)

Related reference:

Chapter 30, "Utility Control Facility (DFSUCF00)," on page 369

DBBBATCH procedure (System Definition)

DLIBATCH procedure (System Definition)

Control statement for the DFSURRL0 utility

The HISAM Reorganization Reload utility has one optional control statement, OPTIONS.

Use the OPTIONS control statement to control whether the DFSURRL0 utility:

- Creates a dump in the event of an abnormal termination
- Generates load statistics for each execution

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.

Examples for the DFSURRL0 utility

The following figure shows an example of the JCL for the DFSURRL0 utility.

Figure 72. Example JCL for the DFSURRL0 utility

```
//HSRELOAD EXEC PGM=DFSRRC00,PARM='ULU,DFSURRL0,,,01,,,,,,,,Y,N'
//*********************************************************************/
//* EXECUTE DATA BASE REORG RELOAD FOR DIVNTZ04                       */
//*********************************************************************/
//IMS      DD DSN=IMSVS.DBDLIB,DISP=SHR
//         DD DSN=I11X.DBDLIB,DISP=SHR
//RECON1   DD DSN=IMSVS.RECON1,DISP=SHR
//RECON2   DD DSN=IMSVS.RECON2,DISP=SHR
//RECON3   DD DSN=IMSVS.RECON3,DISP=SHR
//SYSPRINT DD SYSOUT=A
//DFSUIN01 DD DSN=DATAOUT1,DISP=(OLD,KEEP),UNIT=SYSDA,VOL=SER=USER02
//DBHVSAM1 DD DSN=DIVNTZ04.JJXXS01K,DISP=SHR
//DBHVSAM2 DD DSN=DIVNTZ04.JJXXS01E,DISP=SHR
//DFSVSAMP DD *
2048,4
IOBF=(8192,4)
/*
```

Chapter 27. HISAM Reorganization Unload utility (DFSURUL0)

Use the HISAM Reorganization Unload utility (DFSURUL0) to unload and reorganize HISAM databases and create secondary indexes in HISAM databases.

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 (DFSURRL0).
- Format the index work data sets that the Prefix Resolution utility (DFSURG10) creates so they can be used by the HISAM Reorganization Reload utility (DFSURRL0). The HISAM Reorganization 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 DFSURUL0 blocks the output 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.

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). You can then reload the databases by using the HISAM Reorganization Reload utility (DFSURRL0). 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.

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

The following figure is a flow diagram of this utility.

© Copyright IBM Corp. 1974, 2015

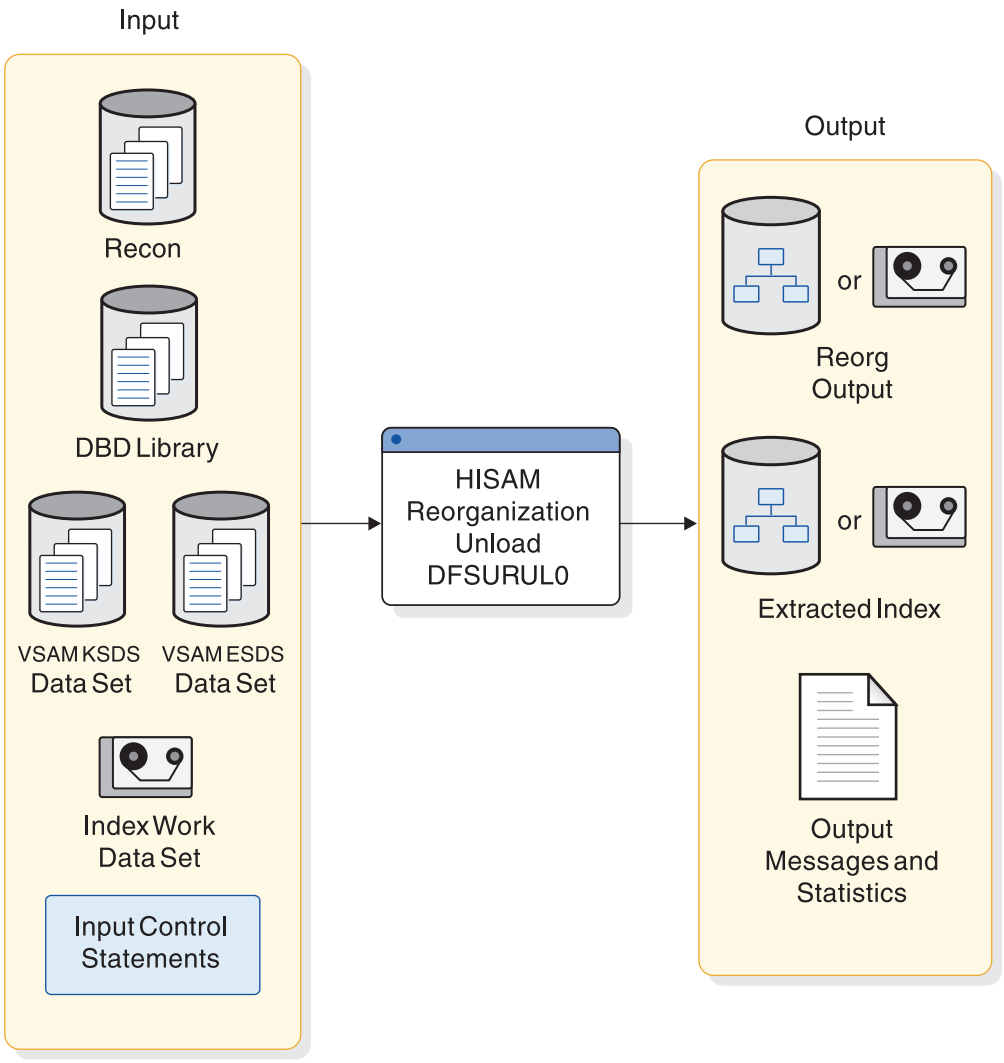


Figure 73. HISAM Reorganization Unload utility (DFSURRL0)

Subsections:

- "Restrictions"
- "Prerequisites" on page 329
- "Requirements" on page 329
- "Recommendations" on page 329
- "Input and output" on page 329
- "JCL specifications" on page 333
- "Return codes" on page 335

Restrictions

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

- This utility cannot unload a PSINDEX database. The HD Reorganization Unload utility (DFSURGU0) is used to unload the PSINDEX database.

- The HISAM Reorganization Unload utility cannot make structural changes or changes to database organizations. Use the HD Reorganization Unload (DFSURGU0) and HD Reorganization Reload (DFSURGL0) 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.
- Do not use this utility to unload or reload a shared secondary index that has alias names that are not registered to DBRC. If you do, IMS issues message DFS194W to indicate that the database referenced by the PSB is not registered with DBRC. IMS then treats the database as if exclusive use was specified for the subsystem by the user.
- Sequential buffering does not support this utility.
- The DFSURUL0 utility works on full-function non-HALDB databases only.
- Utilities that alter databases cannot be run while the database is quiesced.

Prerequisites

Currently, no prerequisites are documented for the DFSURUL0 utility.

Requirements

The requirements for the DFSURUL0 utility include:

- To use this utility for recovery purposes, you must use the HISAM Reorganization Reload utility (DFSURRL0) to reload the database before applying changes to the database. If you do not immediately reload the database before 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, you must perform recovery before you run this utility. 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.
- A 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.

Recommendations

Currently, no recommendations are documented for the DFSURUL0 utility.

Input and output

The primary input to the DFSURUL0 utility is a VSAM KSDS or ESDS data set to be reorganized. The primary output of the DFSURUL0 utility is a reorganized VSAM KSDS or ESDS unload data set.

The following table identifies inputs and outputs for the DFSURUL0 utility.

Table 19. Data sets used by the DFSURUL0 utility

Input	Output
RECON	Output messages and statistics

Table 19. Data sets used by the DFSURUL0 utility (continued)

Input	Output
DBD library	Extracted index
VSAM KSDS data set	Reorganization output
VSAM ESDS data set	
Index work data set	
Input control statements	

Example output report

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. The following figure 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
```

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

Following the message for each data set group, the heading DATA SET GROUP STATISTICS appears.

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 SEGMENT LEVEL STATISTICS section contains the following fields.

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.

JCL specifications

The DFSURUL0 utility is run as a standard z/OS job. The JCL specifications for the DFSURUL0 utility include a JOB statement, the EXEC statement, and the DD statements. One or more utility control statements must be included with the JCL statements.

The following JCL statements are required:

- A JOB statement that you define to meet the specifications of your installation.
- An EXEC statement
- DD statements that define inputs and outputs

EXEC statement

The EXEC statement for the DFSURUL0 utility uses a specific format.

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.

DD statements

The DFSURUL0 utility uses a number of required and optional DD statements.

The DFSURUL0 utility uses the following 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. If you are creating a secondary index, 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. To register the unload data set as an image copy with DBRC, you must reload the database by using the HISAM Reload utility.

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. 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.

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.

Return codes

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

The return codes are as follows:

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

Related concepts:

IMS buffer pools (System Definition)

Related reference:

Chapter 30, “Utility Control Facility (DFSUCF00),” on page 369

Chapter 24, “HD Reorganization Unload utility (DFSURGU0),” on page 291

DBBBATCH procedure (System Definition)

DLIBATCH procedure (System Definition)

Control statements for the DFSURUL0 utility

The DFSURUL0 utility has two control statements to define its processing options: CHANGE and OPTIONS.

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

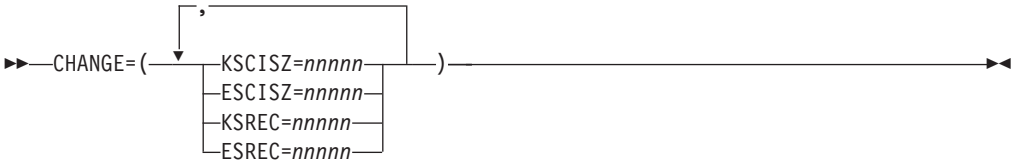
data set. If this field contains the ddname, a corresponding DD statement must be provided. This field must be blank if position 2 contains a 1.

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.

Examples for the DFSURUL0 utility

These examples show usage for the DFSURUL0 utility.

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 the figure below to the sample JCL:

Figure 74. DD statements for using DBRC without dynamic allocation

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

Subsections:

- “Index work data set ” on page 339
- “Extract, replace, and merge index records” on page 339
- “Create and merge secondary indexes, replace constants, and prepare for a HISAM Reorganization Reload utility usage” on page 340

Index work data set

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.

Extract, replace, and merge index records

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
/*
```

Create and merge secondary indexes, replace constants, and prepare for a HISAM Reorganization Reload utility usage

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 28. MSDB-to-DEDB Conversion utility (DBFUCDB0)

Use the MSDB-to-DEDB Conversion utility (DBFUCDB0) 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 DBFUCDB0 utility works with the DEDB Unload/Reload utilities included with IBM IMS Fast Path Basic Tools; however you can use an equivalent alternative product. If you use another product, you must modify the conversion and fallback exit routines that are provided with the DBFUCDB0 utility.

Subsections:

- "Restrictions"
- "Prerequisites"
- "Requirements"
- "Recommendations"
- "Input and output" on page 344
- "JCL specifications" on page 344
- "Return codes" on page 345

Restrictions

The DBFUCDB0 utility has the following restrictions to its use:

- Only non-terminal-related MSDBs can be converted to DEDBs.
- Only one MSDB can be converted for each execution of the DBFUCDB0 utility.
- Each MSDB is converted to a single DEDB area.
- Each MSDB segment is converted to a level '01' segment.
- No consistency checking of fields occurs between the MSDB and the DEDB DBD.
- No checking occurs for the proper ascending order of the RAP RBA values output by the DBFUCDB0 utility in unloaded format.
- The DBFUCDB0 utility will work on Fast Path (DEDB) databases only.
- Utilities that alter databases cannot be run while the database is quiesced.

Prerequisites

Currently, no prerequisites are documented for the DBFUCDB0 utility.

Requirements

If you use an unload and reload product other than the DEDB Unload/Reload utilities delivered with the IMS Fast Path Basic Tools, you must modify the conversion (DBFUCDX0) and fallback (DBFUCDX1) exit routines of the DBFUCDB0 utility so that they work with that unload and reload product.

Recommendations

Currently, no recommendations are documented for the DBFUCDB0 utility.

© Copyright IBM Corp. 1974, 2015

Input and output

The primary input to the DBFUCDB0 utility is non-terminal-related MSDB. The primary output of the DBFUCDB0 utility is a DEDB database containing one area. The DBFUCDB0 utility also produces a summary report.

The following table shows the inputs to and the outputs from the DBFUCDB0 utility:

Table 20. Input to and output from the DBFUCDB0 utility

Input	Output
Non-terminal-related MSDB	DEDB containing one area
Control statements	Summary report

Example output report

This utility generates a summary report when it completes execution. The report is sent to the SYSPRINT data set. The following figure 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
```

JCL specifications

The DBFUCDB0 utility is executed as a standard z/OS job. The JCL specifications for the DBFUCDB0 utility include a JOB statement, the EXEC statement, and the DD statements. One or more utility control statements must be/can be included with the JCL statements.

EXEC statement

The EXEC statement for the DBFUCDB0 utility is specified in the JCL by using a specific format.

Use the following format:

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

DD statements

The DBFUCDB0 utility uses a number of required and optional DD statements.

The DBFUCDB0 utility DD statements include:

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

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.

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.

SYSPRINT DD
Defines the output data set containing error messages, return codes, and the summary report for this utility.

Return codes

One of more of the following return codes are generated by the DBFUCDB0 utility at termination:

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 reference:

z/OS: SORTWKdd DD statement

Control statements for the DBFUCDB0 utility

Control statements for the DBFUCDB0 utility are CONVERT and FALLBACK.

The following list describes the utility control statements:

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.

Examples for the DBFUCDB0 utility

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

Subsections:

- "Converting an MSDB to a DEDB"
- "Reverting a DEDB to an MSDB" on page 348

Converting an MSDB 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 DEDB Unload/Reload utilities (part of the IBM IMS Fast Path Basic Tools for z/OS).

The output data set DBT.FORMAT.SAVE1 will be used as input to the DEDB Unload/Reload utilities (part of the IMS Fast Path Basic Tools) to load the new DEDB area. The new DEDB area is SAVE1.

The following figure is an example of an MSDB being converted to a DEDB.

```
//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
```

Reverting a DEDB to an MSDB

The first step in the fallback procedure is to unload the DEDB using the DEDB Unload/Reload utilities (part of the IMS Fast Path Basic Tools).

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.

The following figure is an example of a DEDB being converted back to an MSDB.

```
//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
```

Related reference:

Chapter 20, “MSDB Dump Recovery utility (DBFDBDR0),” on page 247

Converting an MSDB to a DEDB

Conversion requires the use of the MSDB Dump Recovery utility (DBFDBDR0) and either the DEDB Unload/Reload utilities in the IBM IMS Fast Path Basic Tools for z/OS or an equivalent utility.

Tip: 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.

To convert an MSDB to a DEDB:

1. Make the following DBD changes to the MSDB DBD:
   a. Change the ACCESS parameter in the MSDB DBD from MSDB to DEDB.
   b. Specify the RMNAME parameter for the randomizer name.
   c. Specify an AREA statement for the DEDB area.
2. Create a DEDB DBD for the MSDB to be converted.
3. Run the Database Description Generation utility and the Application Control Blocks Generation utility to create a new application control block library.
4. Run the MSDB Dump Recovery utility to create the MSDBINIT data set. The MSDBINIT data set contains the MSDB segments that are loaded into the new DEDB area.
5. Run the MSDB-to-DEDB Conversion utility with the CONVERT option specified. This utility creates a data set in unload format for each MSDB that is converted.
6. Sort the data set in unload format for each MSDB by ascending order of the RAP RBA values.
7. Run the IMS Fast Path DEDB Reload utility to load these data sets into a DEDB.

Related tasks:

"Performing fallback from a DEDB to an MSDB"

Performing fallback from a DEDB to an MSDB

Fallback requires use of the MSDB Maintenance utility (DBFDBMA0) and a DEDB unload/reload utility (such as the DEDB Unload/Reload utilities in the IBM IMS Fast Path Basic Tools for z/OS).

To perform fallback:

1. Run an unload utility to unload the DEDB. This utility creates the input for the MSDB-to-DEDB Conversion utility.
2. Run the MSDB-to-DEDB Conversion utility with the FALLBACK option specified. The FALLBACK option creates the MSDBLCHG data set, which contains change records that are input to the MSDB Maintenance utility.
3. Sort the MSDBLCHG data set by the root key. The format of the MSDBLCHG data set record is identical to that of the MSDBINIT data set record. The SORT control statements that are 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. The MSDBINIT can then be used to reload the MSDB into storage.

Related tasks:

"Converting an MSDB to a DEDB"

Exit routines for conversion and fallback for the DBFUCDB0 utility

The MSDB-to-DEDB conversion utility (DBFUCDB0) uses exit routines to convert MSDB records into a DEDB unload format and, in the event of a fallback, back into MSDBINIT format from the DEDB unload format.

DEDB unload formats are specific to the utility or tool that you use to unload and reload DEDB databases. The DBFUCDX0 and DBFUCDX1 exit routines shipped with IMS are coded to support the DEDB Unload/Reload utilities included in the separately sold IBM IMS Fast Path Basic Tools for z/OS. If you use a different product to unload and reload your DEDB databases, you must modify both of the DBFUCDX0 and the DBFUCDX1 exit routines to format the MSDB record to a format that your product supports.

Conversion exit routine (DBFUCDX0)

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.

Fallback exit routine (DBFUCDX1)

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. Bind the new DBFUCDX0 into the conversion utility load module.
3. Prepare JCL to execute an unload/reload utility.

Use the same procedure for the FALLBACK option. The new exit routine must be called DBFUCDX1.

Chapter 29. Partial Database Reorganization utility (DFSPRCT1 and DFSPRCT2)

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

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.

Up to 49 related databases can be processed and 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 and up to 10 TOAREAs are allowed after each FROMAREA or KEYRANGE.

You can use the Database Surveyor utility (DFSPRSUR) to produce a report that describes the physical organization of the database. This information can help you determine a range for the reorganization.

The Partial Database Reorganization utility, in one execution, performs operations that are 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:

1. The DFSPRCT1 utility performs preorganization functions to check for user errors without requiring use of the database.
2. The DFSPRCT2 utility performs the unload/reload/pointer resolution functions with the database offline.

The Partial Database Reorganization utility has restart capabilities. Utility 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 (DFSBBO00) to undo any changes made after the checkpoint was taken.

Subsections:

- "Restrictions" on page 354
- "Prerequisites" on page 354
- "Requirements" on page 354
- "Recommendations" on page 354
- "Input and output" on page 354
- "JCL specifications" on page 355

© Copyright IBM Corp. 1974, 2015

• "Return codes" on page 359

Restrictions

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.
- The DFSPRCT1 and DFSPRCT2 utilities will work on full-function non-HALDB databases only.
- Utilities that alter databases cannot be run while the database is quiesced.

Prerequisites

Currently, no prerequisites are documented for the Partial Database Reorganization utility.

Requirements

Currently, no requirements are documented for the Partial Database Reorganization utility.

Recommendations

Currently, no recommendations are documented for the Partial Database Reorganization utility.

Input and output

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

Several reports are produced by the utility.

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 bound before Step 2.) 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, and 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.

JCL specifications

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 to meet the specifications of your installation.
- An EXEC statement
- DD statements that define inputs and outputs

JCL statements for step 1: DFSPRCT1

This section explains the EXEC statement and DD statements for Step 1.

EXEC statement for step 1

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

```
PGM=DFSPRCT1,PARM=()
```

DD statements for step 1

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.

JCL statements for step 2: DFSPRCT2

This section explains the EXEC statement and DD statements for Step 2.

EXEC statement for step 2

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.

DD statements for step 2

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 z/OS SORT/MERGE program. This DD statement is required.

SORTWKnn DD
Defines intermediate storage data sets for the z/OS 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 the z/OS SORT/MERGE program. 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.

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

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.

Return codes

The DFSPRCT1 and DFSPRTC2 modules generate the following 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:

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

Related reference:

Chapter 4, “Database Surveyor utility (DFSPRSUR),” on page 35

Chapter 14, “Batch Backout utility (DFSBBO00),” on page 179

DBBBATCH procedure (System Definition)

DLIBATCH procedure (System Definition)

Control statements for the Partial Database Reorganization utility

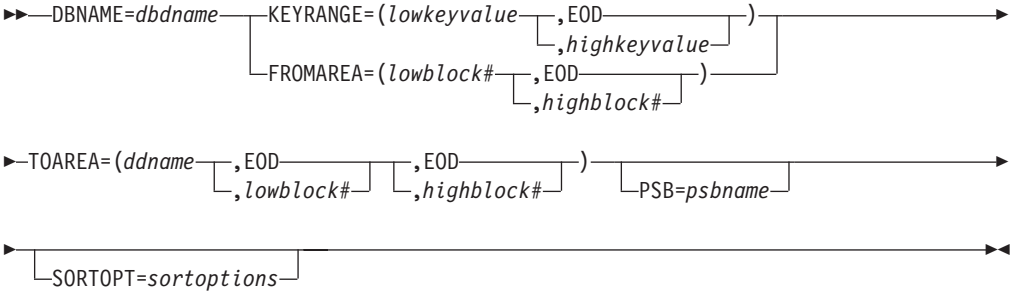
Input statements are used to describe processing options for Partial Database Reorganization utility.

Input statements must conform to the following guidelines:

- 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.

DFSPRCT1 utility control statement for step 1



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.

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.

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.

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.

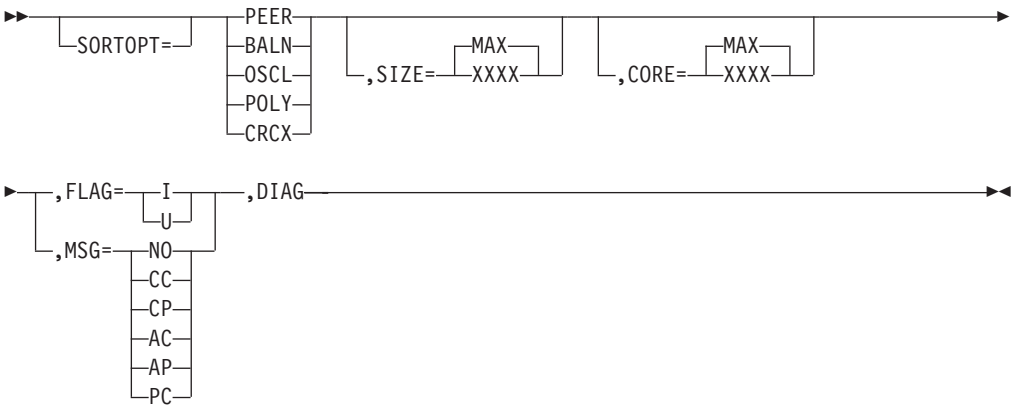
SORTOPT

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

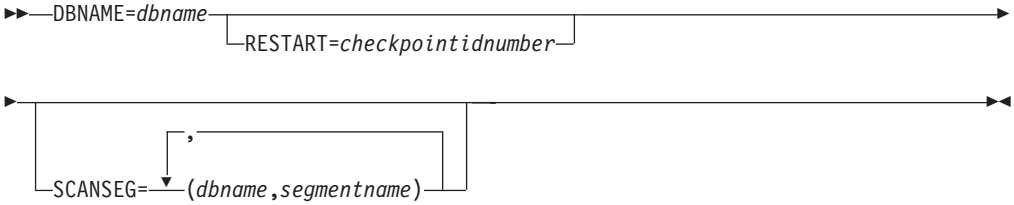
Restriction: The character string located in quotes cannot exceed 69 characters.

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:



DFSPRCT2 utility control statement for step 2



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.)

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

Examples for the Partial Database Reorganization utility

The following examples show sample JCL for the DFSPRCT1 and DFSPRCT2 modules.

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 the figure below 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
```

Subsections:

- "Running DFSPRCT1"
- "Running DFSPRCT2" on page 364

Running DFSPRCT1

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. The following figure 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
```

The following figure 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
```

The following figure 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
```

The following figure 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
```

PSB example

The following figure 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
```

DBD examples

The following figure 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
```

Running DFSPRCT2

The following figure 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
/*
```

The following are sample output reports from Step 2. The following figure 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
```

The following figure contains 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'
```

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
```

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
```

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.

The following figure shows 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
```

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

The following figure 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
```

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)

The following figure 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
```

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 30. Utility Control Facility (DFSUCF00)

Use the Utility Control Facility (UCF) to control the execution and options of the IMS database reorganization utilities.

The UCF is a utility that controls the execution of other utilities. The UCF is driven by control statements coded in a free-form manner. You can specify multiple control statements to run multiple utilities (such as execution of the various reorganization and batch Image Copy utilities) on the same or multiple databases in the same job step.

The UCF provides the following advantages:

- 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 and 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 runs all functions in a manner that protects you from certain operational problems.

Subsections:

- "Restrictions"
- "Prerequisites" on page 370
- "Requirements" on page 370
- "Recommendations" on page 372
- "JCL specifications" on page 372
- "Return codes" on page 377

Restrictions

The following restrictions apply to the Utility Control Facility:

- The UCF does not support the following utilities:
  - The Change Accumulation utility
  - The Database Backout utility
  - The Database Image Copy 2 utility
  - The Database Recovery utility
  - The Database Surveyor utility
  - The Online Database Image Copy utility
  - The Partial Database Reorganization utility
- The UCF does not support HALDB databases.
- Do not use the UCF to initially load a GSAM database.
- FUNCTION=ZB does not update RECON when DBRC is active.
- The DFSUCF00 utility works on full-function non-HALDB databases only.
- Utilities that alter databases cannot be run while the database is quiesced.

© Copyright IBM Corp. 1974, 2015

Prerequisites

Currently, no prerequisites are documented for the DFSUCF00 utility.

Requirements

The Utility Control Facility has the following requirements:

General requirements

- 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.
- 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.
- The FUNCTION=OP card must have the same checkpoint value and request parameters as the FUNCTION=OP card in the failing UCF step.
- If the system abruptly terminates, due to a loss of power, for example, 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.

Load and reload requirements

- Observe the following precautions for the restart of your 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 non-unique, 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 non-unique and a root segment insert is done for a segment that 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 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.
- 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.
- The FUNCTION=OP card must have the same checkpoint value and request parameters as the FUNCTION=OP card in the failing UCF step.
- If the system abruptly terminates, due to loss of power, for example, 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.
- For restart of your 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 your 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.
- 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.

A region size of 600 KB is the minimum required for execution.

For restart processing, specify PARM='ULU,DFSUCF00,,,0001'.

Recommendations

Currently, no recommendations are documented for the DFSUCF00 utility.

Input and output

DFSUCF00 executes other utility programs by reading *xx*SYSIN DD statements. The outputs are the results of executing the utilities themselves.

JCL specifications

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

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.

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.

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 before 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.

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 z/OS 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 z/OS SORT/MERGE program. The “nn” designation is a two-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. The following table summarizes the use of the FUNCTION keyword on the various DD statements.

Table 21. 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									
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.

Keyword summary tables

The following table 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 22. 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
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												

Table 22. 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**

Key:

blank	A blank space means that 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.

The following table shows the minimum keywords that you must specify when constructing each type of control statement.

Table 23. 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

Upon termination of the UCF, the return codes in the following table are passed to register 15.

Table 24. UCF return codes

Code	Meaning	Action
0	Processing was normal.	DFSNJRNL and DFSNCDS data sets can be scratched.
4	Warning messages have been issued, or termination was caused because EXEC=STOP was specified.	Verify that warnings are not errors; if no errors exist take same action as for return code 0. If restart is wanted because of errors or EXEC=STOP, the DFSNJRNL and DFSNCDS must have been saved.

Table 24. UCF return codes (continued)

Code	Meaning	Action
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.

Related concepts:

Tailoring the IMS system to your environment (System Definition)

Overview of the IMS system definition process (System Definition)

Related reference:

Chapter 22, "HALDB Migration Aid utility (DFSMAID0)," on page 267

Control statements for the DFSUCF00 utility

Control statements for the DFSUCF00 utility 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 initial load of a database, or both.

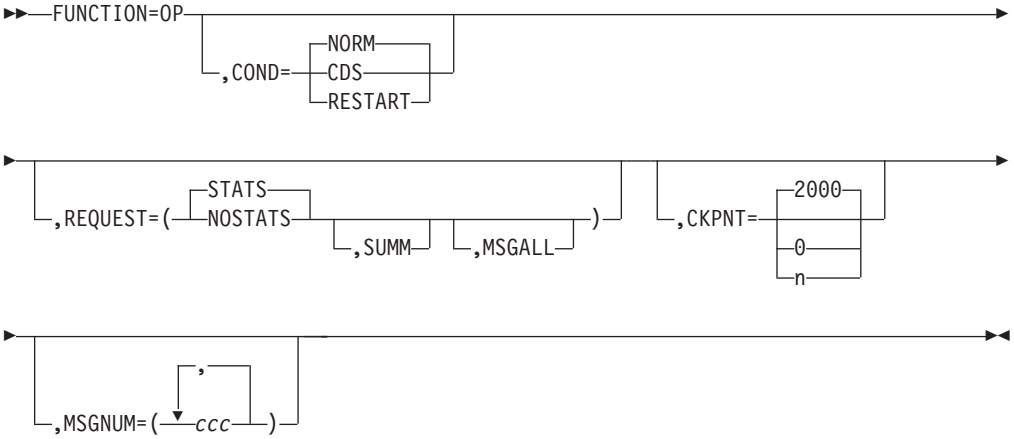
Subsections:

- "FUNCTION=OP statement" on page 380
- "FUNCTION=DR statement" on page 381
- "FUNCTION=DU statement" on page 382
- "FUNCTION=DX statement" on page 384
- "FUNCTION=IL statement" on page 386
- "FUNCTION=IM statement" on page 387
- "FUNCTION=PR statement" on page 389
- "FUNCTION=PU statement" on page 390
- "FUNCTION=RR statement" on page 392
- "FUNCTION=RU statement" on page 393
- "FUNCTION=SN statement" on page 396
- "FUNCTION=SR statement" on page 398
- "FUNCTION=SU statement" on page 400
- "FUNCTION=SX statement" on page 402
- "FUNCTION=ZB statement" on page 404
- "FUNCTION=ZM statement" on page 406

FUNCTION=OP statement

Because UCF takes the place of the Database Prereorganization utility, the REQUEST= options from this statement is used by the Database Prefix Resolution to control the report writing. Therefore, code the REQUEST keyword primarily for Prereorganization/Prefix Resolution, and secondarily for UCF defaults.

The format of the statement is:



This control statement establishes certain UCF run specifications.

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.

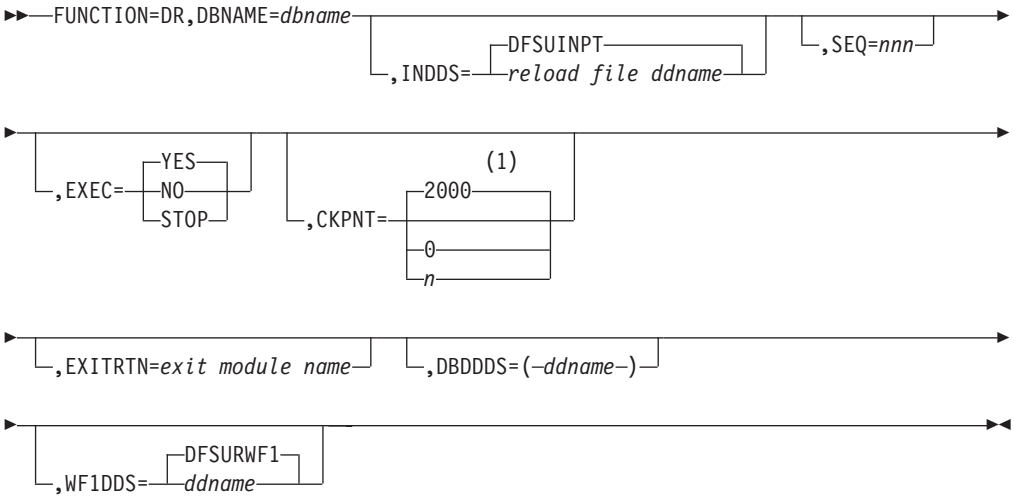
FUNCTION=DR statement

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.

DBDDS=
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.

FUNCTION=DU statement

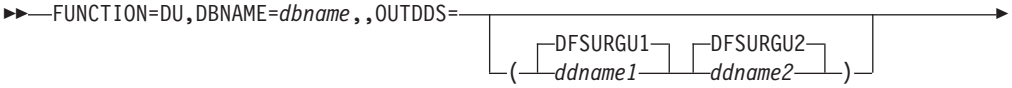
This control statement causes the UCF to execute the HD Reorganization Unload utility to unload an HDAM or HIDAM database to a QSAM-formatted data set. When the DBD has both changed and contains logical relationships, UCF execution must be stopped after the unload function if one of the following conditions exist:

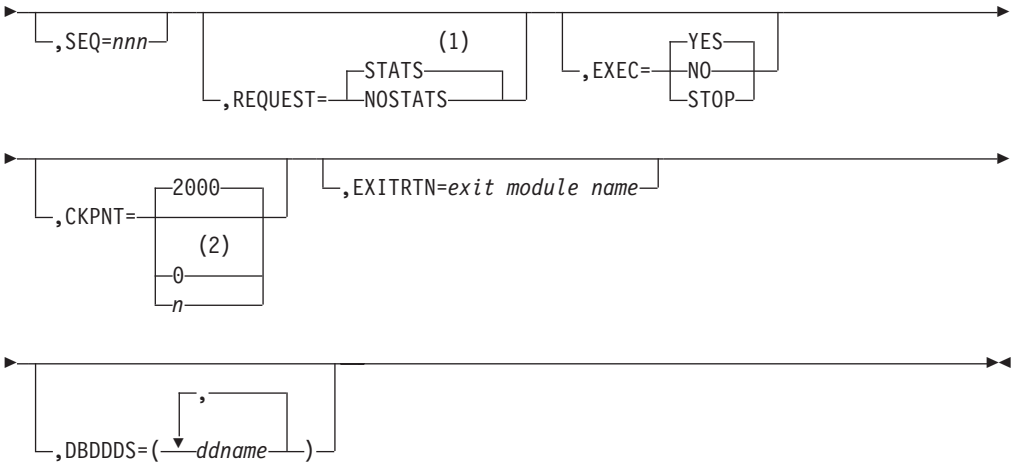
- Either counter, LT, or LP pointers are changed.
- The level of segments involved in logical relationships change from adding or deleting segments.
- The DBD name is changed and the DBD contains logical relationships.

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:





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 “Requirements” on page 370.

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.

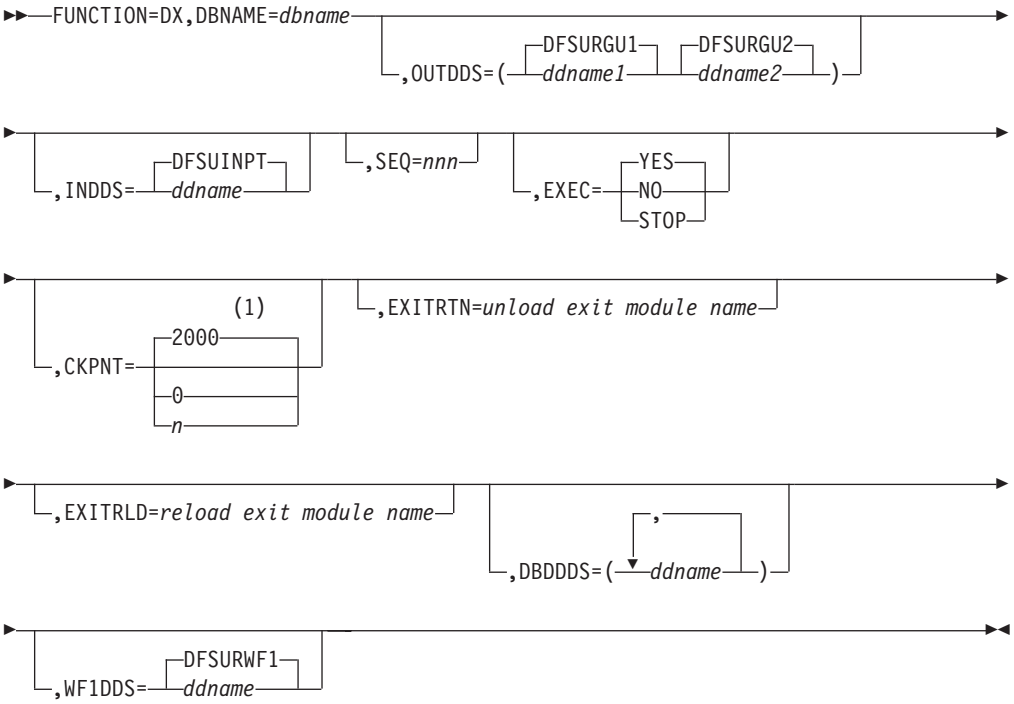
DBDDDS=
Allows you to verify that all database data sets are defined before executing this function. The UCF executes a DEVTYPE macro against any ddnames; if there is not a DD statement for any of the ddnames specified, a message is issued and restart preparation begins to terminate the UCF.

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 “Requirements” on page 370.

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.

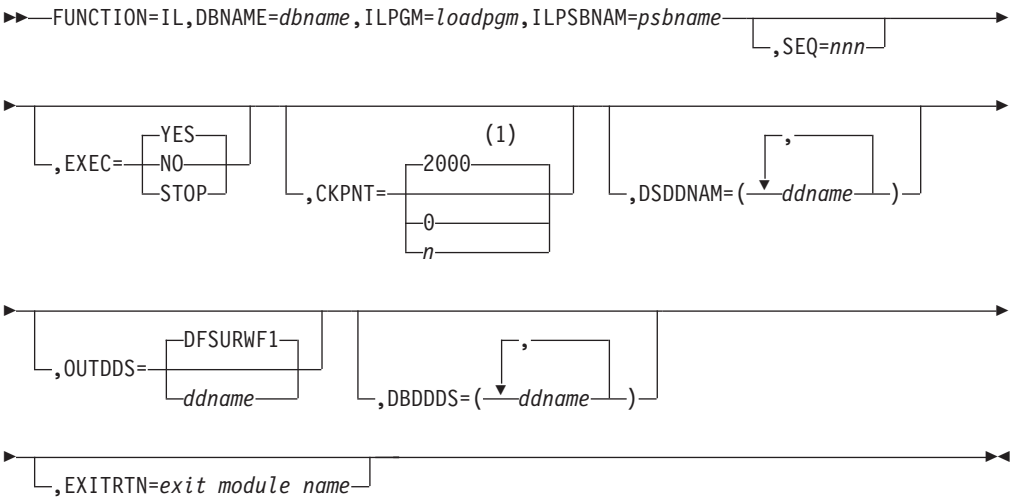
WF1DDS=
Defines the output work data set that is supplied as an input to the Prefix Resolution utility. The data set can reside on a tape or a direct-access device. If a ddname other than the default (DFSURWF1) is used, this ddname must be consistent across all references to this data set within an execution of the UCF.

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.

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 BLDLand ATTACHcommands.

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.

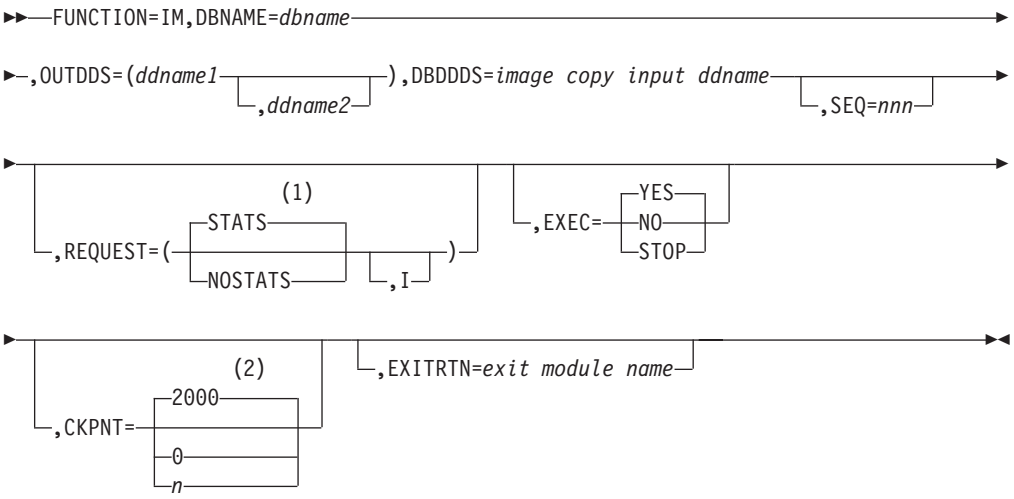
EXITRTN=
Specifies an exit routine to be given control to allow you to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

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.

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.

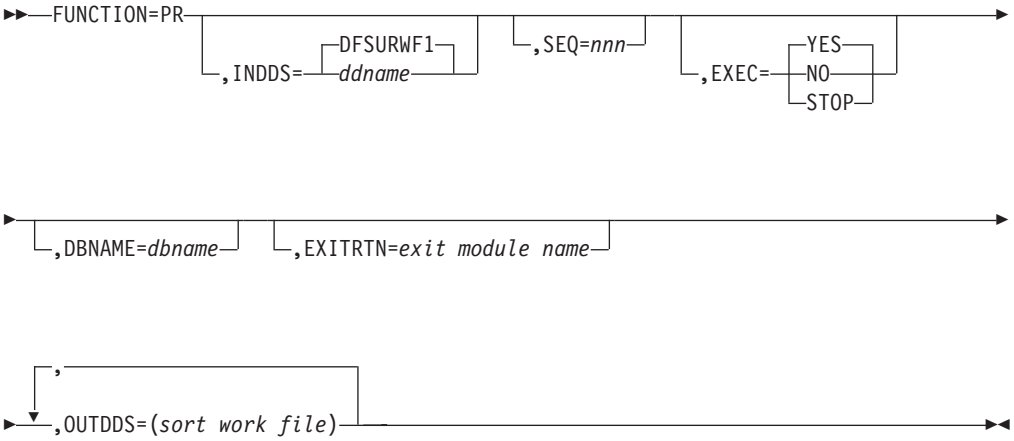
The content of this data set can be different when compared to the same data set generated by the Prefix Resolution utility run without UCF. This can occur if you have logically related databases and database records which are logical parents without logical children. In this case, some pointers will 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 no DD statement for any of the sort-work-files is specified, a message is issued and restart preparation terminates the UCF.

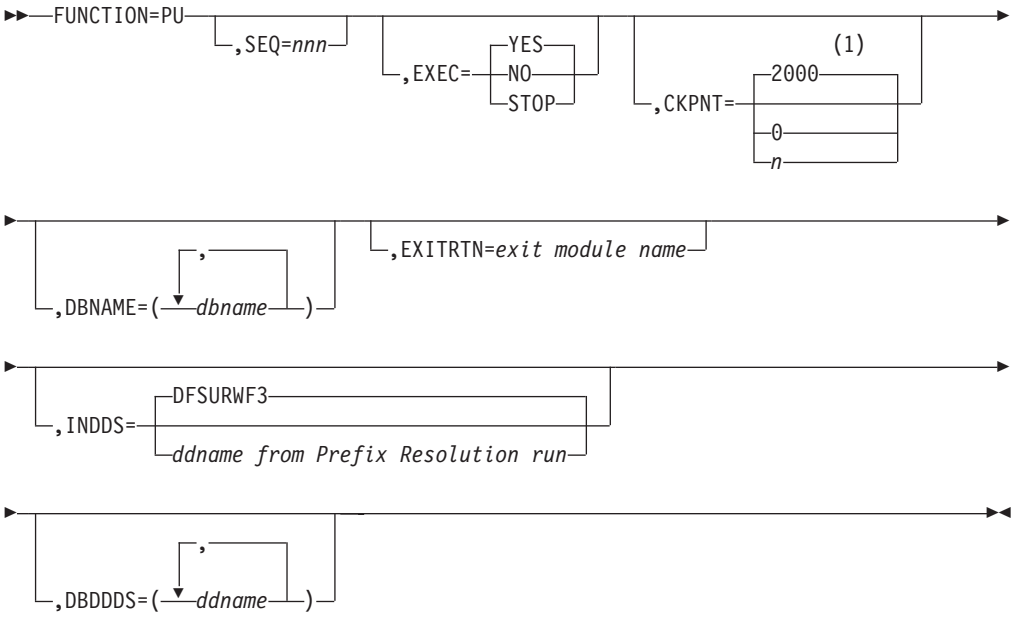
FUNCTION=PU statement

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).

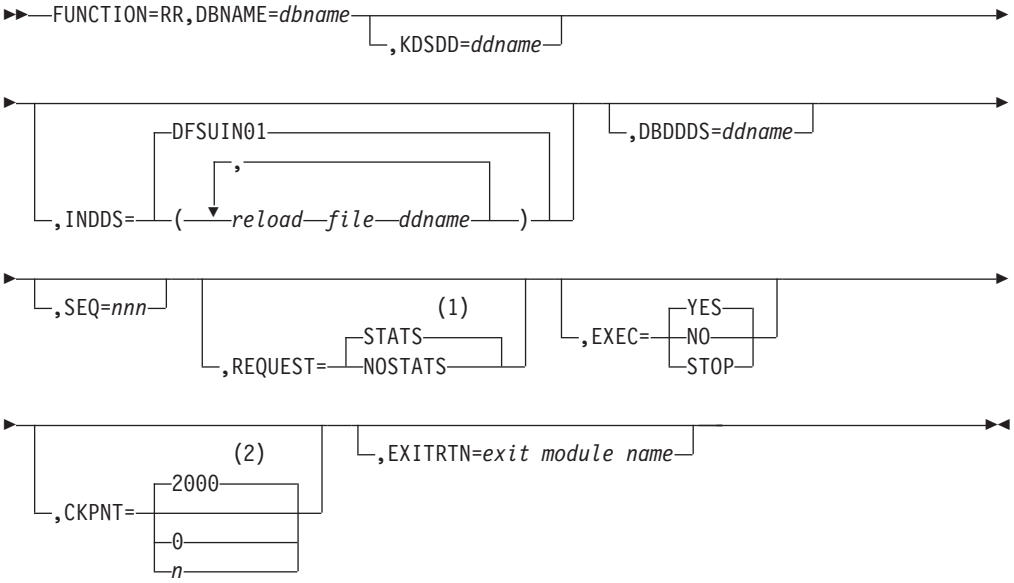
DBDDS=
Allows you to verify that all database data sets are defined before executing this function. The UCF executes a DEVTYPE macro against any ddnames; if there is not a DD statement for any of the ddnames specified, a message is issued and restart processing begins to terminate the UCF.

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.

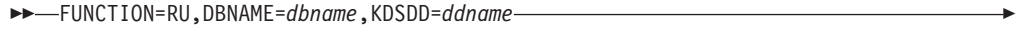
EXITRTN=
Specifies an exit routine to be given control in order to allow you to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

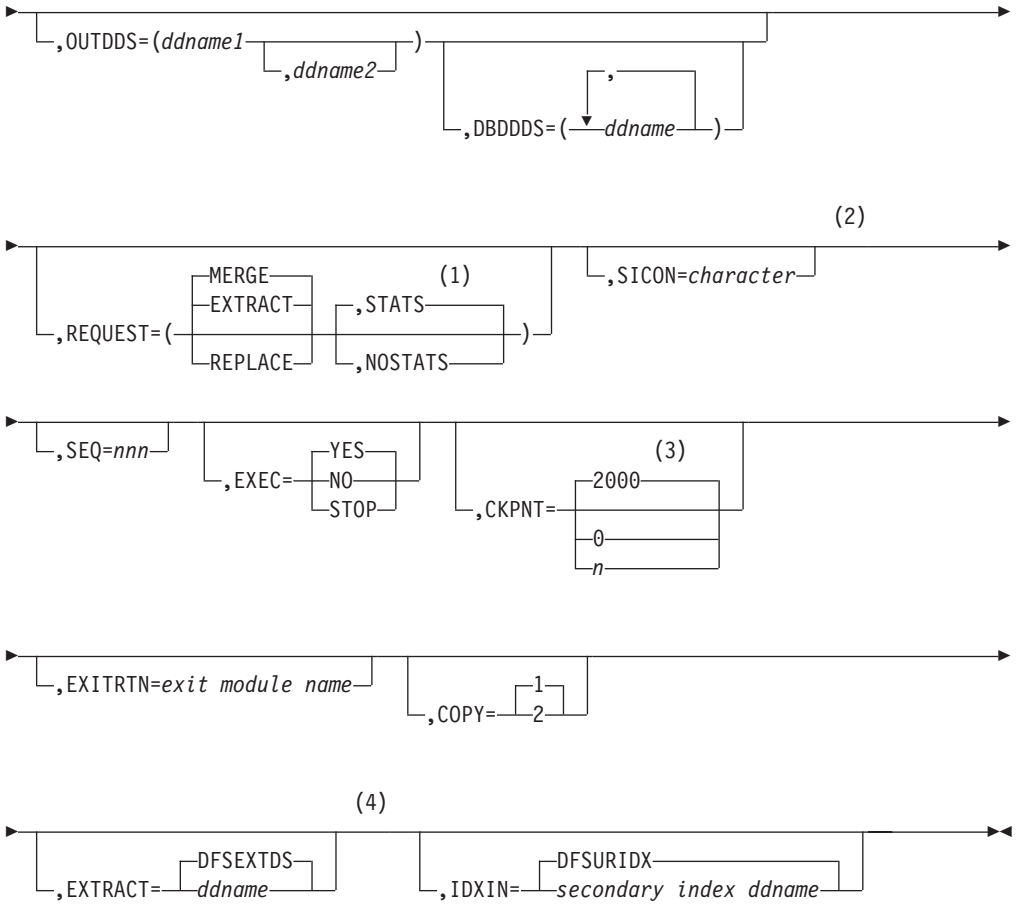
FUNCTION=RU statement

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.

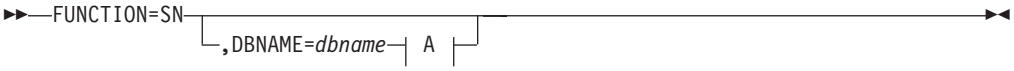
FUNCTION=SN statement

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.

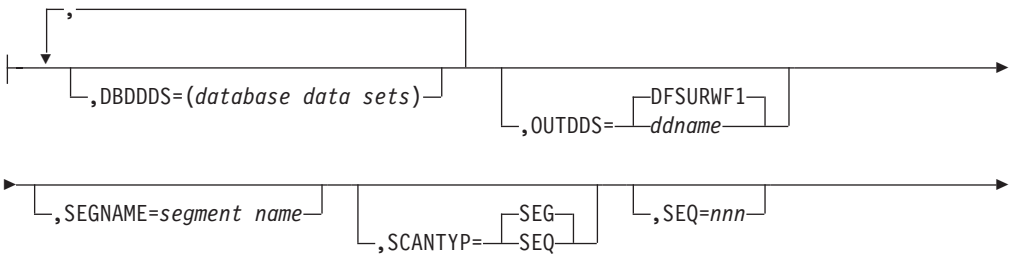
The SCAN function is executed for initial loads and HD Reloads. When SCAN is not required, the DFSURWF1 DD statement must be present unless EXEC=NO is specified on the FUNCTION=SN statement.

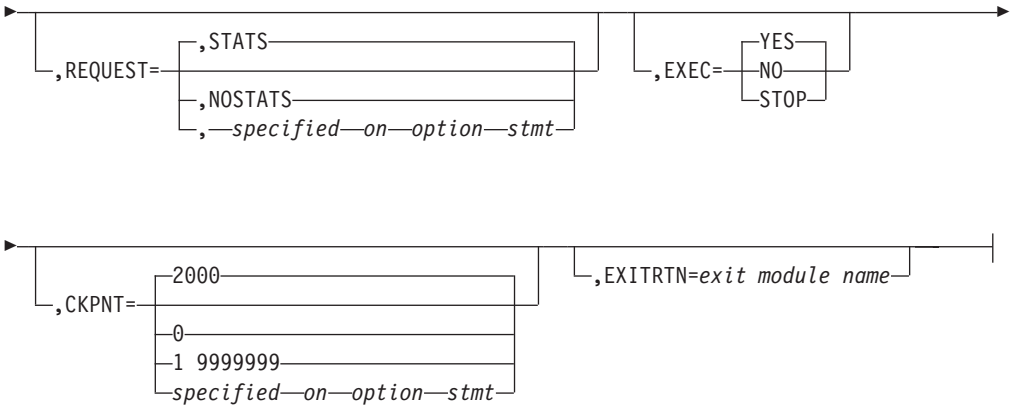
Do not specify more than 20 ddnames for this function.

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.

DBDDS=
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.

FUNCTION=SR statement

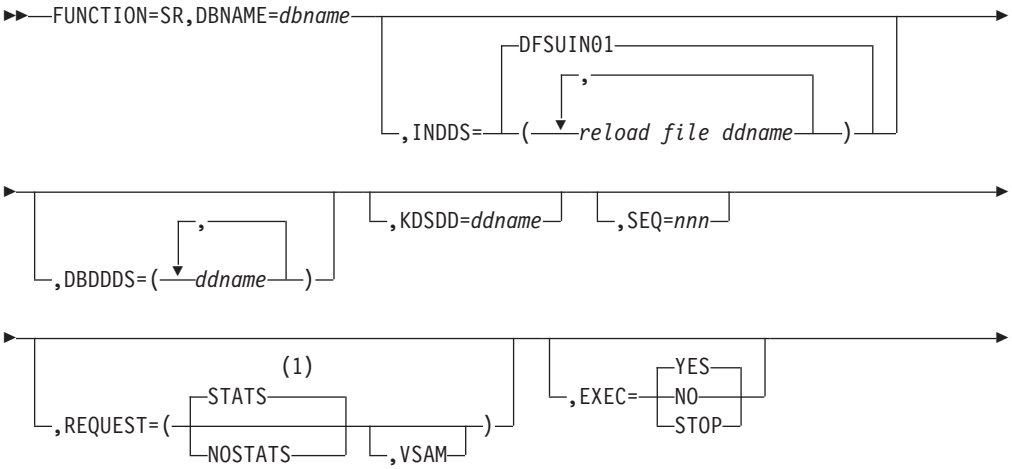
This control statement causes the UCF to execute the HISAM Reorganization Reload utility (DFSURRL0). This utility can be used to reload a HISAM or HIDAM primary index database from a QSAM-formatted data set created by the HISAM Reorganization Unload utility.

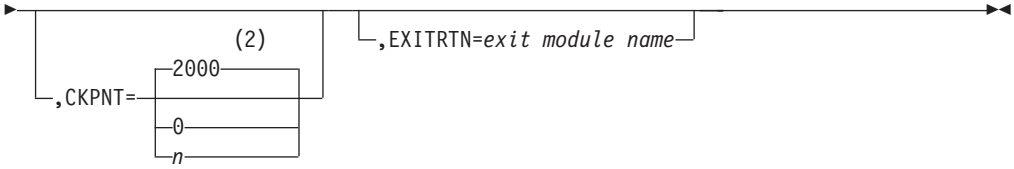
Reorganization of HISAM databases is significantly faster using the HISAM Unload/Reload utilities instead of the HD Unload/Reload utilities.

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.

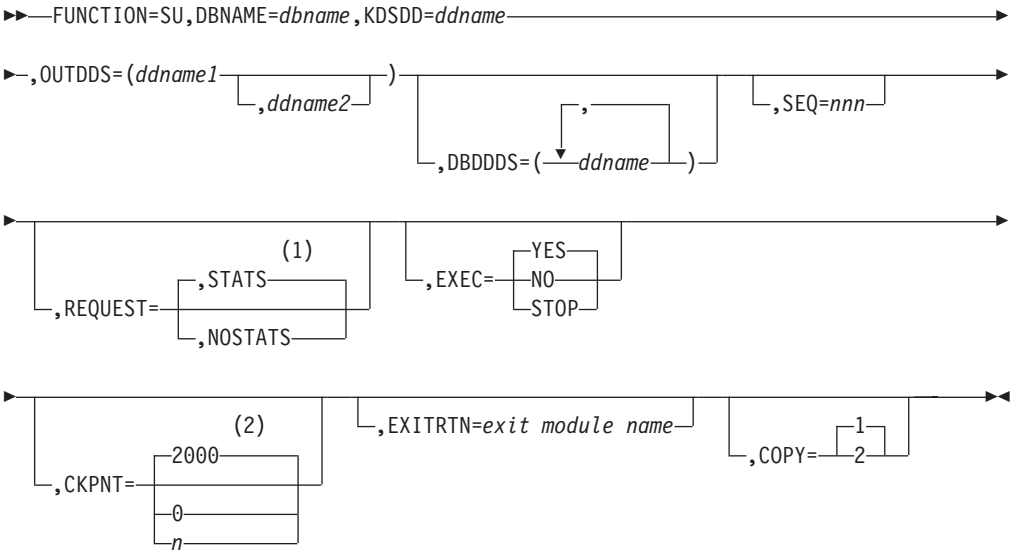
EXITRTN=
Specifies an exit routine to be given control to allow you to examine records or compile statistics. The exit-module-name must reside in a library known to this function.

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. .

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=

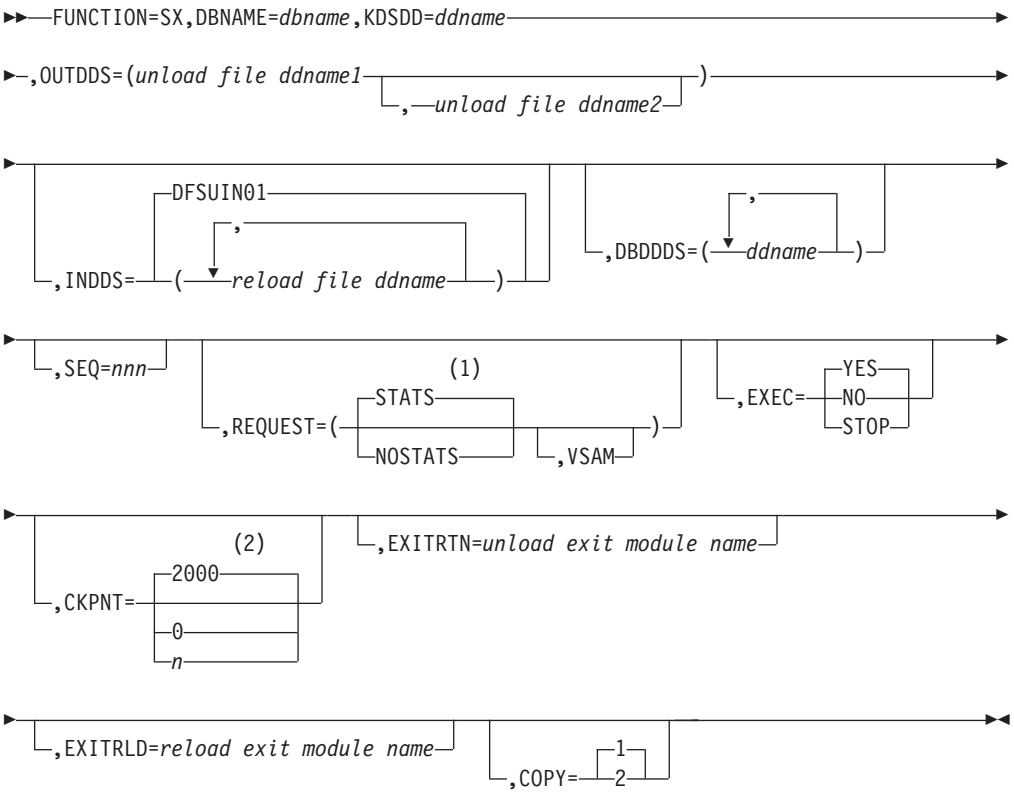
Provides for multiple copies of output data sets. COPY can be specified only once for each control statement. COPY=1 produces only one copy and is the default. COPY=2 produces an additional output copy. Specify it if a ddname-2 is specified for the OUTDDS keyword. (Specifying multiple output copies can be advantageous; if a permanent error occurs on one copy, the remaining volume continues to normal end of job.)

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. .

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.)

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.

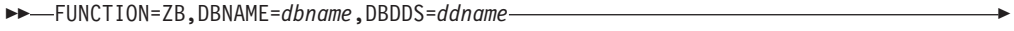
Restriction: This utility function should be used only by an IBM Field Engineering Program Support Representative or by someone under that person's direction.

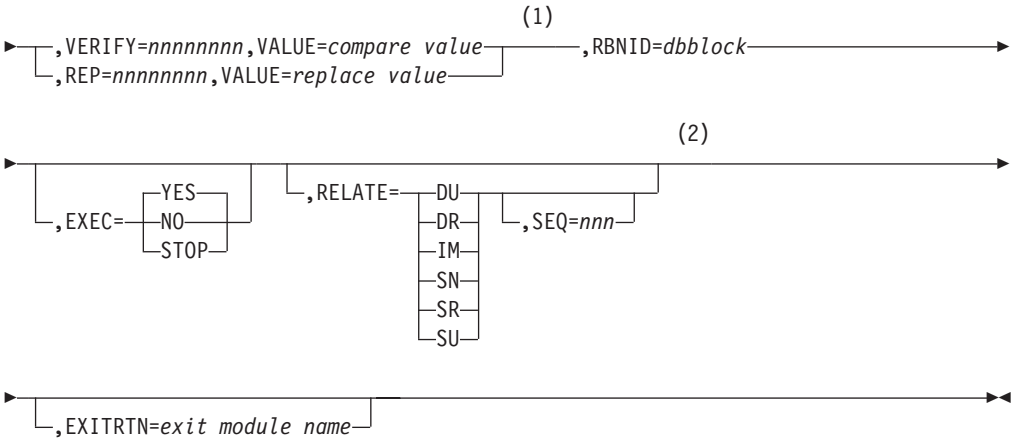
FUNCTION=ZB does not update the RECON when DBRC is active.

Tip: 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.

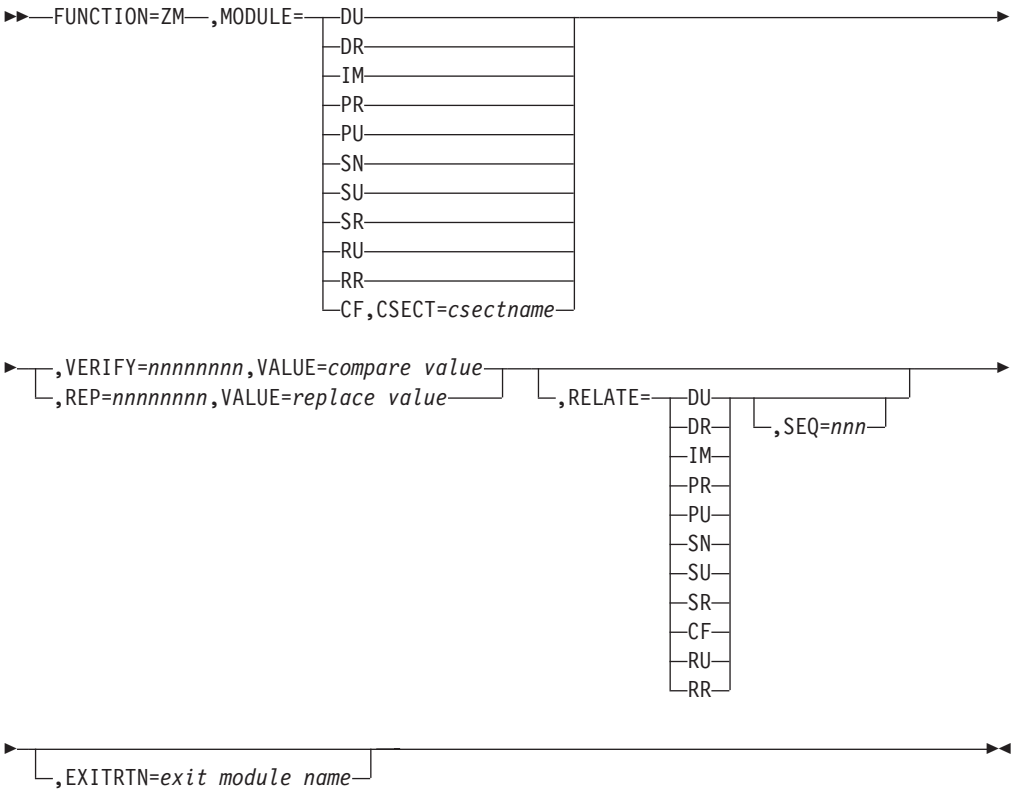
EXITRTN=
Use to obtain control at the time the block is in storage to verify that the zap was performed, and, if desired, to compile statistics on the block.

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.

Examples for the DFSUCF00 utility

These examples show the use of the Utility Control Facility using sample JCL.

Subsections:

- "Minimum JCL required to execute the UCF"
- "JCL to execute UCF in a restart"
- "JCL and a control statement to execute UCF in a restart" on page 410

Minimum JCL required to execute the UCF

The following figure 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.
```

JCL to execute UCF in a restart

The following figure 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.
```

JCL and a control statement to execute UCF in a restart

The following figure 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.
```

Running the DFSUCF00 utility

Running the DFSUCF00 utility has several operational considerations, such as normal processing, termination error processing, restart processing, and user-supplied exit routine processing.

Subsections:

- "Normal processing"
- "Initial load application program considerations" on page 412
- "Termination/error processing" on page 413
- "Restart processing" on page 414
- "User-supplied exit routine processing" on page 415

Normal processing

Normal processing requires control statements for the direction of all utility functions except Database Scan (DFSURGS0), Prefix Resolution (DFSURG10), and Prefix Update (DFSURGP0). The UCF automatically generates all required statements for these three functions except:

- If an unload or reload only is requested, Database Scan, Prefix Resolution, and Prefix Update processing is not automatic; you must provide control statements to request execution of these utilities.
- If only an initial load is requested, processing of these utilities is automatic, unless keywords on their control statements indicate no processing is to take place. One control 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 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

This topic contains General-use Programming Interface information.

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 non-unique 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.

Initial load exit routine

This topic contains General-use Programming Interface information.

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 re-initialized 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.

Termination/error processing

Error checking occurs both during execution of the utility and after completion. If there are no errors, the completion of the event is recorded and a check is made for a request to stop processing. In the event that a request to stop processing was made, restart messages are generated and the job ends with a return code of 4. If no stop-processing request was made, the next function is determined and processing continues as described for normal processing.

If errors are discovered during the error-checking phase of execution, the completion of the event is recorded as an error and processing ends to allow for restart of the function.When the entire job is completed, return codes are passed.

With a normal completion, restart processing is not necessary and is prevented by a special record written on the journal data set. Any termination other than normal can be restarted. Statistics are printed upon termination of each function and at job termination.

A return code of zero in register 15 indicates normal completion of the user's Initial Load program and that a restart is not to be done at a later time. If restart is to be done later, however, register 15 contains a return code greater than 4. This causes a checkpoint to be taken, and the restart proceeds with the next root segment.

Checkpoint/restart

The UCF contains an internal checkpoint/restart feature for abnormal termination and termination requests from the user. The checkpoint function requires a journal data set, and you must allocate this for the UCF. The IMS system log normally used for batch processing is not used in this instance. Checkpoints are taken when a functional utility is started or ended, and when a control function has been started or ended. Checkpoints are also taken at specific points within the processing of a functional utility as a result of user-supplied and default record counts. At each of these checkpoints, records are written to the journal data set to define the type of checkpoint and the appropriate restart control information.

These checkpoints are used by the restart processor and are internal checkpoints rather than z/OS checkpoints. The frequency of checkpoints can be specified as a user option or can be defaulted to every 2000 database related records.

Restart processing

When restart processing is required, the control data set that was in use when the program last ended is read, and the journal log that was last used is also read. The last function started, completed, or both is determined by matching the journal records to the functions in the control data set, and processing continues as in normal processing.

Depending on the type of checkpoint records that were last written to the journal data set, the restart function occurs in one of the following ways:

- If the records indicate the start of a functional utility (that is, one of the database utilities or the user's initial load program), restart processing begins at the function that was started. If the records indicate the end of a functional utility, restart processing begins at the next function to be performed.
- If the records indicate the start or end of a control function (such as building the control data set), restart processing begins either at the control function that was started or at the next function to be performed.
- If the records indicate a checkpoint, restart processing begins at that point, and the IMS data sets are positioned as necessary.

Restart processing requires the availability of all data sets that were in use at the time the checkpoint was taken.

During restart, when repositioning any multiple volume output data set, the DD statement must be changed to remove those volume serial numbers not used in the original execution. If this is not done, the output data set might not be correctly positioned.

Restart of the Prefix Resolution utility must be from the beginning of execution. For this utility, all data sets created up to the point of termination must be scratched and the utility must be restarted as if for the first time.

User-supplied exit routine processing

This topic contains General-use Programming Interface information.

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. This segment will not be included in the reload utility statistics. 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.

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.

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.

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.

Table 25. Database zaps

Before unload utility functions	After load utility functions
IM	SR
SU	DR
RU	RR
DU	RV

All other database zaps are executed after the last requested functional utility has executed.

Write-to-operator-with-reply function

The Write-to-operator-with-reply function (WTOR) issues messages to, and processes replies from, the UCF user. The outstanding WTOR provided by the UCF allows you to query the UCF to determine the status of its execution, to change checkpoint values, to stop the UCF, and then to restart it at a later time.

Restriction: The restart cannot be initiated by a WTOR.

The WTOR is processed between executions of the utilities and at checkpoint time.

In the figure below, you receive message DFS367I at the console, request the status of the UCF's execution, and learn that it is executing the HD Reorganization Reload utility with the database “DBNAME”. (The DBNAME is not included in the message for function PR.)

```
@09   *DFS367I UTILITY CONTROL FACILITY RUNNING,
        ENTER REQUESTS AS NEEDED
r 9, status
IEE600I  REPLY TO 09 IS 'STATUS'
DFS369I  FUNCTION IS DR FOR DATABASE DBDNAME
@10   *DFS367I UTILITY CONTROL FACILITY RUNNING,
        ENTER REQUESTS AS NEEDED
```

In response to the last DFS367I message, you might, for example, request that a checkpoint value be changed immediately as follows:

```
r 10,ckpnt=100
```

Or you might, for example, request the UCF to stop processing with:

```
r 10,END
```

The UCF stops processing upon recognition of the END request by the executing utility.

Additionally, you can enter certain control replies and option replies in response to message DFS367I. See the table below for a summary of information that you can enter.

Table 26. 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.

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.

Related reference:

z/OS: SPZAP control statements

Part 5. Report and test utilities

Use the report and test utilities to report and test databases and their use within IMS.

Each topic introduces one utility, describes how it works, defines the requirements and restrictions for its use, and provides examples.

© Copyright IBM Corp. 1974, 2015

Chapter 31. 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

Each field in the reports is explained, followed by a summary of how you can use the report.

Subsections:

- "Restrictions"
- "Prerequisites"
- "Requirements"
- "Recommendations"
- "Input and output" on page 422
- "JCL specifications" on page 422
- "Return codes" on page 423

Restrictions

The DFSUTR30 utility depends on the data records on the data set produced by the DB Monitor (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.

Prerequisites

Prior to running the DFSUTR30 utility, you must enable the DB Monitor to create the input data set used by the DFSUTR30 utility.

Requirements

Currently, no requirements are documented for the DFSUTR30 utility.

Recommendations

Currently, no recommendations are documented for the DFSUTR30 utility.

© Copyright IBM Corp. 1974, 2015

Input and output

The primary input to the DFSUTR30 utility is the output data set of the DB Monitor. The primary output of the DFSUTR30 utility are any of various reports summarizing database activities in a batch system.

Table 27. Input to and output from the DFSUTR30 utility

Input	Output
IMSMON or IMSLOG data set	Reports
Analysis control data set	

JCL specifications

The DFSUTR30 utility runs in batch mode, with one job step for each monitor trace period.

The following are required:

- A JOB statement that you define to meet the specifications of your installation.
- 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 the DB monitor module DFSMNTB0. It can be a separate data set (ddname= and dsname=IMSMON) or the system log (dsname=IMSLOG).

ANALYSIS DD
Specifies the Analysis Control data set. This file must be in card image format. Use DD DUMMY for default parameters (no distribution report), and input is the first trace interval on the tape.

Analysis control data set

The Analysis Control data set has three record types that allow you to request the Distribution-Appendix report, to redefine distribution intervals, and to specify which trace interval is to be processed.

- To generate the Distribution-Appendix report, specify either DISTRIBUTION or DIS, beginning in column 1.
- To override the default distribution intervals, specify control statements in the form Dn n1,n2,...

- To denote which trace interval (other than the first, which is the default) on the input data set is to be processed, specify FILE=nn, or FILE=n, beginning in column 1.

The FILE= specification does not refer to OS files. It refers to trace intervals recorded within an OS file.

Return codes

Currently, no return codes are documented for the DFSUTR30 utility.

Related concepts:

Monitoring IMS (Operations and Automation)

IMS Monitor reports (System Administration)

Examples for the DFSUTR30 utility

These examples show sample JCL for the DFSUTR30 utility.

The following figure 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
/*
```

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 the following figure:

```
//ANALYSIS DD  *
DISTRIBUTION
FILE=2
D13 ,,2,4,6,8,10,12
```

In the example the first two intervals are not changed, but remain as 0 and 1, respectively.

Chapter 32. Program-Isolation-Trace Report utility (DFSPIRP0)

Use the Program-Isolation-Trace Report utility (DFSPIRP0) to print a report from the X'67FA' log records produced by a program isolation (PI) trace.

A program isolation (PI) trace writes all Program Isolation Lock Manager enqueue and dequeue requests to a trace table. The report created by the DFSPIRP0 utility shows only those enqueues that were required to wait (the resource was not immediately available).

You can restrict the report produced by the DFSPIRP0 utility to a time period by specifying a PRINT control statement.

The following topics provide additional information:

- "Restrictions"
- "Prerequisites"
- "Requirements"
- "Recommendations"
- "Input and output" on page 426
- "JCL specifications" on page 427
- "Return codes" on page 428

Restrictions

Keywords in the DFSPIRP0 utility control statements cannot be repeated.

The DFSPRIRP0 utility will work on full-function (HALDB and non-HALDB) databases only.

Prerequisites

Before running the DFSPIRP0 utility you must make sure that certain prerequisites are met.

Prior to running the DFSPIRP0 utility, you must enable a program isolation (PI) trace to create the X'67FA' log records from which the DFSPIRP0 utility generates a program isolation trace report. You can enable a program isolation trace by issuing the command /TRACE SET ON PI OPTION ALL.

Requirements

To run the DFSPIRP0 utility you must satisfy at least one operational requirement.

For the purposes of sorting the log records, you must define at least three sort work data sets to the DFSPIRP0 utility.

Recommendations

Currently, no recommendations are documented for the DFSPIRP0 utility.

© Copyright IBM Corp. 1974, 2015

Input and output

The primary input to the DFSPIRP0 utility are the log input data sets that contain the X'67FA' log records created by a program isolation trace. The primary output of the DFSPIRP0 utility is a report of the program isolation enqueues that had to wait for a the necessary resource.

The following table shows the input to and output from the DFSPIRP0 utility:

Table 28. Input to and output from the DFSPIRP0 utility

Input	Output
Log input data sets that contain the X'67FA' log records	Report of program isolation enqueues that were required to wait
Control statements	Messages

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 X'67FA' log records 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 required to wait, but its PI trace log record is 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 specifications

The DFSPIRP0 utility is executed as a standard z/OS job. The JCL specifications for the DFSPIRP0 utility include a JOB statement, the EXEC statement, and the DD statements. An optional utility control statement can be included with the JCL statements.

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)
```

SYSDA must refer to a single disk device type because the sort program does not allow the work areas to be on different 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.

Return codes

No return codes are provided by the DFSPIRP0 utility.

Related concepts:

Obtaining program isolation and lock traces (System Administration)

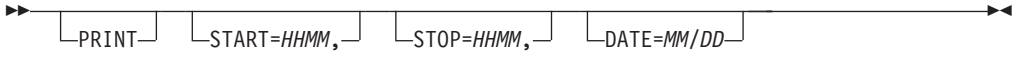
Related reference:

KBLA Knowledge-Based Record Formatting and Print module (DFSKBLAK) (System Utilities)

Control statement for the DFSPIRP0 utility

Use the optional DFSPIRP0 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 of the DFSPIRP0 utility control statement 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.

Examples for the DFSPIRP0 utility

The following examples show sample JCL for using the DFSPIRP0 utility.

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
/*
```

If the report is required to be from 11:30 p.m. until 1:10 a.m. the control statement is:

```
PRINT START=2330,STOP=2510
```

The following figure 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
```

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 can reprocess the SB buffer handler call sequence issued during a previous execution of an application program or another utility.

The SB Test utility can accept as input *SB image capture log records*. You can generate SB image capture log records by including an SBIC control statement in the //DFSCTL file of the JCL for an IMS program, which captures on the IMS log all internal IMS calls to the SB buffer handler during the execution of the program.

The SB Test utility can only process the SB image capture log records from a single execution of a program that uses the same PSB as the utility. If your input data set for this utility contains SB image capture log records from multiple executions of a program using the same PSB, you need to include a SELECT utility control statement to specify which execution of the program you want this utility to process. If you do not include a SELECT statement, the SB Test utility processes the SB image capture log records of only the first execution of the program.

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 include a DBIO utility control statement to specify that it issue DB Read I/Os.

Because the SB Test utility can recreate and re-execute the SB buffer handler call sequence generated by an application, the SB Test utility is 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.

Important: A re-execution of an SB buffer handler call sequence by the SB test utility does not always result in 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.

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 following figure depicts the data set requirements for the SB Test utility.

© Copyright IBM Corp. 1974, 2015

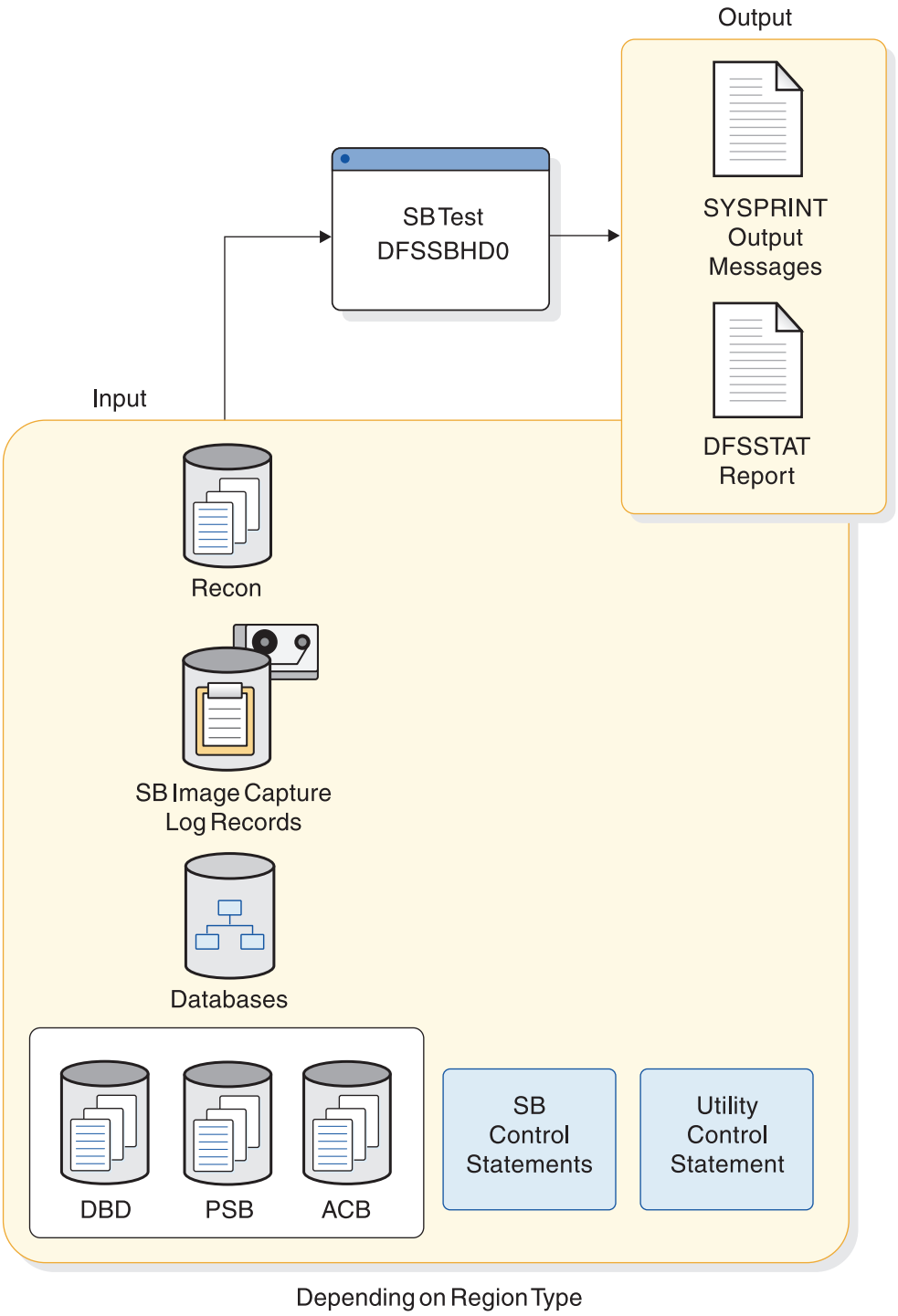


Figure 75. Data set requirements for the SB Test utility

The following topics provide additional information:

- "Restrictions" on page 433
- "Prerequisites" on page 433
- "Requirements" on page 433
- "Recommendations" on page 433
- "Input and output" on page 433
- "JCL specifications" on page 434

- "Return codes" on page 436

Restrictions

The following restrictions apply to the SB Test utility:

- The SB Test utility processes the SB image capture log records of only one program execution each time it is run.
- This utility executes only in batch regions.
- This utility does not process SB image capture log records for DB PCBs that specify a load processing option.
- This utility can simulate only 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.
- The DFSSBHD0 utility will work on full-function (HALDB and non-HALDB) databases only.

Prerequisites

Before you can use SB image capture log records as input to the SB Test utility, you must execute an application program or utility by using JCL that includes an SBIC control statement in the //DFSCTL file.

Requirements

To simulate the SB buffer handler calls of an application or utility, the SB Test utility requires access to:

- The PSB used by the application
- All of the DBDs referenced by the application or utility
- The database data sets accessed by the application or utility

Recommendations

Do not use the IMS Monitor during execution of the SB Test utility, because the IMS Monitor does not provide meaningful reports for the SB Test utility.

Input and output

The following table identifies the inputs and outputs used by the SB Test utility.

Table 29. 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	

The SB Test utility uses the following input:

- A SYSUT1 data set containing the SB image capture log records.

- An optional SYSIN file containing the DBIO and the SELECT utility control statements.
- 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 DFSSTAT file containing SB summary and SB detail reports.
- A SYSPRINT file listing the utility control statements that were read from SYSIN, along with messages written by the utility.

JCL specifications

The SB Test utility is executed as a standard z/OS job. The following statements are required:

- A JOB statement that you define to meet the specifications of your installation.
- An EXEC statement
- DD statements defining inputs and outputs

EXEC statement

The EXEC 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'
  ```

Specify in the *psbname* parameter the name of the PSB used by the application that generated the SB image capture log records. Specify in the *dbdname* parameter the name of the DBD that 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*.

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.

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.

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

Return codes

Currently, no return codes are documented for the DFSSBHD0 utility.

Related concepts:

IMS buffer pools (System Definition)

Related reference:

Sequential buffering control statements (System Definition)

DBBBATCH procedure (System Definition)

DLIBATCH procedure (System Definition)

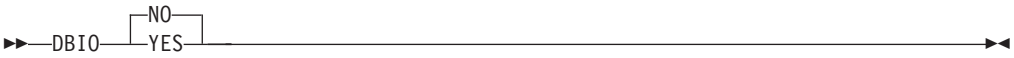
Control statements for the DFSSBHD0 utility

The DFSSBHD0 utility uses two control statements: DBIO and SELECT.

Use the DBIO statement to specify whether the SB Test utility uses real DB read I/O calls or simulated read I/O calls. Use the SELECT statement to select a specific program execution when the SB image capture log includes multiple program executions.

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.

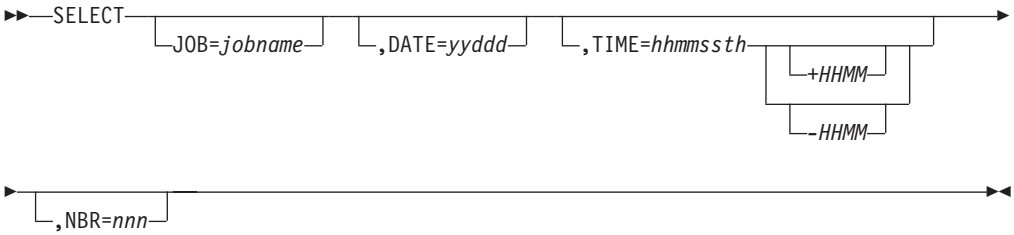
When YES is specified, only those OSAM Read I/O operations issued by the OSAM buffer handler are issued. I/O operations to VSAM database data sets and output DB I/O operations are not issued during execution of this utility.

Important: 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.

Examples for the DFSSBHD0 utility

These examples shows sample JCL for the DFSSBHD0 utility.

The following figure shows JCL for execution of the SB Test utility.

Figure 76. JCL for execution of the SB Test utility

```
//    EXEC  DLIBATCH,MBR=DFSSBHD0,PSB=xxxxxxxx
//SKILLDB   DD DSN=user.db.data.dsname,DISP=SHR
//SKILLIX   DD DSN=user.db.indx.dsname,DISP=SHR
//SYSUT1    DD DSN=imslog,DISP=SHR
//DFSVSAMP  DD *
4096,6
//DFSCTL    DD *
SBPARM ACTIV=COND,BUFSETS=10
//SYSIN     DD *
DBIO YES
//DFSSTAT   DD SYSOUT=A
//SYSPRINT  DD SYSOUT=A
/*
```

Part 6. Appendixes

© Copyright IBM Corp. 1974, 2015

Notices

This information was developed for products and services offered in the U.S.A.

This material may be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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, MD-NC119
Armonk, NY 10504-1785
United States of America

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, 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.

© Copyright IBM Corp. 1974, 2015

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites 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 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. The sample programs are provided "AS IS," without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

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.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Programming interface information

This information documents Product-sensitive Programming Interface and Associated Guidance 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 this software product. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. 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 may need to be changed in order to run with new product releases or versions, or as a result of service. Product-sensitive Programming Interface and Associated Guidance Information is

identified where it occurs, either by an introductory statement to a section or topic, or by a Product-sensitive programming interface label. IBM requires that the preceding statement, and any statement in this information that refers to the preceding statement, be included in any whole or partial copy made of the information described by such a statement.

Privacy policy considerations

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, See IBM's Privacy Policy at http://www.ibm.com/privacy and IBM's Online Privacy Statement at http://www.ibm.com/privacy/details the section entitled "Cookies, Web Beacons and Other Technologies" and the "IBM Software Products and Software-as-a-Service Privacy Statement" at http://www.ibm.com/software/info/product-privacy.

Bibliography

This bibliography lists all of the publications in the IMS Version 11 library, supplemental publications, publication collections, and accessibility titles cited in the IMS Version 11 library.

IMS Version 11 library

Title	Acronym	Order number
IMS Version 11 Application Programming	APG	SC19-2428
IMS Version 11 Application Programming APIs	APR	SC19-2429
IMS Version 11 Commands, Volume 1: IMS Commands A-M	CR1	SC19-2430
IMS Version 11 Commands, Volume 2: IMS Commands N-V	CR2	SC19-2431
IMS Version 11 Commands, Volume 3: IMS Component and z/OS Commands	CR3	SC19-2432
IMS Version 11 Communications and Connections	CCG	SC19-2433
IMS Version 11 Database Administration	DAG	SC19-2434
IMS Version 11 Database Utilities	DUR	SC19-2435
IMS Version 11 Diagnosis	DGR	GC19-2436
IMS Version 11 Exit Routines	ERR	SC19-2437
IMS Version 11 Installation	INS	GC19-2438
IMS Version 11 Licensed Program Specifications	LPS	GC19-2439
IMS Version 11 Master Index and Glossary	MIG	SC19-2440
IMS Version 11 Messages and Codes, Volume 1: DFS Messages	MC1	GC19-4232
IMS Version 11 Messages and Codes, Volume 2: Non-DFS Messages	MC2	GC19-4233
IMS Version 11 Messages and Codes, Volume 3: IMS Abend Codes	MC3	GC19-4234
IMS Version 11 Messages and Codes, Volume 4: IMS Component Codes	MC4	GC19-4235
IMS Version 11 Operations and Automation	OAG	SC19-2441
IMS Version 11 Release Planning	RPG	GC19-2442
IMS Version 11 System Administration	SAG	SC19-2443
IMS Version 11 System Definition	SDG	GC19-2444
IMS Version 11 System Programming APIs	SPR	SC19-2445
IMS Version 11 System Utilities	SUR	SC19-2446

Supplementary publications

Title	Order number
IMS and SOA Executive Overview	GC19-2516
Program Directory for Information Management System Transaction and Database Servers V11.0	GI10-8788
IMS Version 11 Fact Sheet	GC19-2451
IRLM Messages and Codes	GC19-2666

© Copyright IBM Corp. 1974, 2015

Publication collections

Title	Format	Order number
IMS Version 11 Product Kit	CD	SK5T-7375

Accessibility titles cited in the IMS Version 11 library

Title	Order number
z/OS TSO/E Primer	SA22-7787
z/OS TSO/E User's Guide	SA22-7794
z/OS ISPF User's Guide Volume 1	SC34-4822

Index

Special characters

%DFSHALDB (HALDB Partition Definition utility)
  batch JCL requirements
    DD statements 85
  examples 86, 88
  export 89
  foreground JCL requirements
    DD statements 84
  import 89
  input and output 82
  interactive application 89
  overview 81
  prerequisites 82
  recommendations 82
  requirements 82
  restrictions 81
  return codes 86
  utility control statements 87

A

ABEND control statement
  Database Prefix Update utility (DFSURGP0) 19
ABEND parameter
  Database Prereorganization utility (DFSURPR0) 264
  HISAM Reorganization Reload utility (DFSURRL0) 325
  HISAM Reorganization Unload utility (DFSURUL0) 338
ABEND statement
  Batch Backout utility (DFSBBO00) 188
  Database Recovery utility (DFSURDB0) 221
  Database Scan utility (DFSURGS0) 32
ABENDMSG statement
  Batch Backout utility (DFSBBO00) 189
ABENDOFF parameter
  Database Prereorganization utility (DFSURPR0) 264
  HISAM Reorganization Reload utility (DFSURRL0) 325
  HISAM Reorganization Unload utility (DFSURUL0) 338
accessibility
  features ix
  keyboard shortcuts ix
action statement
  MSDB Maintenance utility (DBFDBMA0) 121
ACTIVE statement
  Batch Backout utility (DFSBBO00) 189
asynchronous read ahead 315

B

back up databases 129
backup
  Database Image Copy 2 utility (DFSUDMT0) 143
  Database Image Copy utility (DFSUDMP0) 131
  Online Database Image Copy utility (DFSUICP0) 169
Batch Backout utility (DFSBBO00)
  description 179
  example 192
  in an RSR environment 193
  input and output 182
  JCL requirements
    DD statements 185
    EXEC statement 185
  prerequisites 181
  QUIESCE database status 180
  recommendations 181
  Remote Site Recovery 193
  requirements 181
  restrictions 180
  return codes 183
  utility control statements
    ABEND statement 188
    ABENDMSG statement 189
    ACTIVE statement 189
    BYPASS LOGVER statement 189
    BYPASS SEQVER statement 190
    CHKPT statement 190
    COLDSTART statement 191
    NOREADBACK statement 191
    overview 188
    READBACK statement 192
buffer sets 315
buffers
  BUFNO command 315
  DBFUHDR0 (High-Speed DEDB Direct Reorganization utility) 315
  High-Speed DEDB Direct Reorganization utility (DBFUHDR0) 315
BYPASS LOGVER statement
  Batch Backout utility (DFSBBO00) 189
BYPASS SEQVER statement
  Batch Backout utility (DFSBBO00) 190

C

CHANGE= statement
  HISAM Reorganization Unload utility (DFSURUL0) 337
CHKPT statement
  Batch Backout utility (DFSBBO00) 190
  Database Prefix Update utility (DFSURGP0) 18
CHKPT statement *(continued)*
  Database Scan utility (DFSURGS0) 31
CKPNT= keyword
  control statements
    UCF FUNCTION=DR 382
    UCF FUNCTION=DU 384
    UCF FUNCTION=DX 385
    UCF FUNCTION=IL 387
    UCF FUNCTION=IM 389
    UCF FUNCTION=OP 380
    UCF FUNCTION=PU 391
    UCF FUNCTION=RR 393
    UCF FUNCTION=RU 395
    UCF FUNCTION=SN 398
    UCF FUNCTION=SR 400
    UCF FUNCTION=SU 401
    UCF FUNCTION=SX 404
COLDSTART statement
  Batch Backout utility (DFSBBO00) 191
concurrent copy function
  DBDS select statement 151
configuration
  HALDB (High Availability Large Database) 114
control statements
  %DFSHALDB (HALDB Partition Definition utility) 87
  Batch Backout utility (DFSBBO00)
    overview 188
  Database Change Accumulation utility (DFSUCUM0)
    overview 201
  Database Image Copy 2 utility (DFSUDMT0)
    overview 151
  Database Image Copy utility (DFSUDMP0) 140
  Database Prefix Update utility (DFSURGP0)
    overview 18
  Database Prereorganization utility (DFSURPR0)
    overview 262
  Database Recovery utility (DFSURDB0)
    overview 221
  Database Scan utility (DFSURGS0)
    overview 30
  Database Surveyor utility (DFSPRSUR) 38
  DBFDBDR0 (MSDB Dump Recovery utility) 251
  DBFDBMA0 (MSDB Maintenance utility) 121
  DBFUCDB0 (MSDB-to-DEDB Conversion utility) 346
  DBFUMIN0 (DEDB Initialization utility) 52

© Copyright IBM Corp. 1974, 2015

control statements *(continued)*
DEDB Initialization utility (DBFUMIN0) 52
DFSBBO00 (Batch Backout utility)
overview 188
DFSMAID0 (HALDB Migration Aid utility) 271
DFSPIRP0 (Program-Isolation-Trace Report utility) 428
DFSPRCT1 and DFSPRCT2 (Partial Database Reorganization utility) 359
DFSPREC0 (HALDB Index/ILDS Rebuild utility) 244
DFSPRSUR (Database Surveyor utility) 38
DFSSBHD0 (SB Test utility)
overview 436
DFSUCF00 (Utility Control Facility)
overview 378
DFSUCUM0 (Database Change Accumulation utility)
overview 201
DFSUDMP0 (Database Image Copy utility) 140
DFSUDMT0 (Database Image Copy 2 utility)
overview 151
DFSUICP0 (Online Database Image Copy utility) 174
DFSUPNT0 (HALDB Partition Data Set Initialization utility) 79
DFSURDB0 (Database Recovery utility)
overview 221
DFSURGL0 (HD Reorganization Reload utility) 283
DFSURGP0 (Database Prefix Update utility)
overview 18
DFSURGS0 (Database Scan utility)
overview 30
DFSURGU0 (HD Reorganization Unload utility) 301
DFSURPR0 (Database Prereorganization utility)
overview 262
DFSURRL0 (HISAM Reorganization Reload utility) 325
DFSURUL0 (HISAM Reorganization Unload utility)
overview 336
HALDB Index/ILDS Rebuild utility (DFSPREC0) 244
HALDB Migration Aid utility (DFSMAID0) 271
HALDB Partition Data Set Initialization utility (DFSUPNT0) 79
HALDB Partition Definition utility (%DFSHALDB) 87
HD Reorganization Reload utility (DFSURGL0) 283
HD Reorganization Unload utility (DFSURGU0) 301
HISAM Reorganization Reload utility (DFSURRL0) 325
control statements *(continued)*
HISAM Reorganization Unload utility (DFSURUL0)
overview 336
MSDB Dump Recovery utility (DBFDBDR0) 251
MSDB Maintenance utility (DBFDBMA0) 121
MSDB-to-DEDB Conversion utility (DBFUCDB0) 346
Online Database Image Copy utility (DFSUICP0) 174
Partial Database Reorganization utility (DFSPRCT1 and DFSPRCT2) 359
Program-Isolation-Trace Report utility (DFSPIRP0) 428
SB Test utility (DFSSBHD0)
overview 436
Utility Control Facility (DFSUCF00)
overview 378
COPY= keyword
control statements
UCF FUNCTION=RU 396
UCF FUNCTION=SU 402
UCF FUNCTION=SX 404
CSECT= keyword
UCF FUNCTION=ZM control statement 407

D

data sets
concatenated data sets
displaying concatenated HALDB data sets 113
Database Image Copy 2 utility (DFSUDMT0)
dynamic data set names 164
DFSURDB0 SNAP output data set 219
HALDB
displaying concatenated data sets 113
DATA= keyword
MSDB Maintenance utility MSDBINIT statement 124
Database Change Accumulation utility (DFSUCUM0)
description 195
examples 208
input and output 197
JCL requirements
DD statements 199
EXEC statement 198
prerequisites 197
purge date and time 202
QUIESCE database status 195
recommendations 197
requirements 197
restricted from Utility Control Facility 369
restrictions 196
return codes 201
utility control statements
DB0 statement 205
DB1 statement 206
ID statement 204
Database Change Accumulation utility (DFSUCUM0) *(continued)*
utility control statements *(continued)*
overview 201
SO statement 207
Database Image Copy 2 utility (DFSUDMT0)
concurrent copy function 160
copy complete notification 164
data sets, multiple input 161
description 143
DFSMSdss SET PATCH commands 166
dynamic data set names 164
examples 150
fast replication function 161
group names 162
input and output 145
JCL requirements
DD statements 147
EXEC statement 146
prerequisites 144
QUIESCE database status 143
recommendations 145
requirements 144
restrictions 143
same data set option 163
utility control statements
concurrent copy DBDS select statement 151
fast replication DBDS select statement 153
group name statement 155
HLQ specification statement 157
overview 151
SET PATCH control statement 159
Database Image Copy utility (DFSUDMP0)
description 131
examples 140
in an RSR environment 134
input and output 135
JCL requirements
DD statements 138
EXEC statement 135
JCL specifications 135
prerequisites 133
QUIESCE database status 132
Remote Site Recovery 134
requirements 133
restrictions 132
return codes 135, 146
utility control statements 140
Database Prefix Resolution utility (DFSURG10)
example JCL 11
EXEC statement example 7
EXEC statement PARM field options 7
input and output 6
introduction 3
JCL example 11
JCL requirements
DD statements 9
EXEC statement 7
prerequisites 5

Database Prefix Resolution utility (DFSURG10) *(continued)*
QUIESCE database status 4
recommendations 6
requirements 5
restrictions 4
return codes 6
z/OS SORT/MERGE program 7
Database Prefix Update utility (DFSURGP0)
example 19
input and output 15
introduction 13
JCL requirements
DD statements 16
EXEC statement 16
prerequisites 15
QUIESCE database status 14
recommendations 15
requirements 15
restart 24
restrictions 14
return codes 15
utility control statements
ABEND 19
CHKPT 18
overview 18
RSTRT 18
SNAP 19
Database Prereorganization utility (DFSURPR0)
description 257
example 264
JCL requirements
DD statements 260
EXEC statement 260
output messages 259
QUIESCE database status 258
restrictions 258
return codes 262
utility control statements
DBIL= 262
DBR= 263
OPTIONS= 263
overview 262
Database Recovery utility (DFSURDB0)
description 211
DFSUSNAP DD statement 219
examples 224
input and output 215
JCL requirements
DD statements 218
EXEC statement 218
prerequisites 213
QUIESCE database status 213
recommendations 215
Remote Site Recovery 211
requirements 213
restrictions 213
return codes 220
RSR environment 211
SNAP output data set attributes 219
utility control statements
ABEND 221
NOSEQCK 221
overview 221
S (database recovery) 221
Database Scan utility (DFSURGS0)
description 25
example JCL 32
input and output 27
JCL requirements
DD statements 28
EXEC statement 28
JCL specifications and examples 27
output messages 27
prerequisites 27
QUIESCE database status 27
recommendations 27
requirements 27
restart 33
restrictions 27
return codes 27
scan options
SEG 30
SEQ 30
utility control statements
ABEND 32
CHKPT 31
DBS 30
overview 30
RSTRT 31
Database Surveyor utility (DFSPRSUR)
description 35
examples 40
input 35
JCL requirements
DD statements 37
EXEC statement 36
output 36
prerequisites 35
QUIESCE database status 35
recommendations 35
requirements 35
restricted from Utility Control Facility 369
restrictions 35
return codes 36
utility control statements 38
database zap capability
Utility Control Facility (DFSUCF00) 416
Database-Monitor Report Print utility (DFSUTR30)
analysis control data set 422
description 421
example 423
input and output 422
JCL requirements
DD statements 422
EXEC statement 422
JCL specifications 422
prerequisites 421
recommendations 421
requirements 421
restrictions 421
return codes 423
databases
backing up
Database Image Copy 2 utility (DFSUDMT0) 143
Database Image Copy utility (DFSUDMP0) 131
databases *(continued)*
backing up *(continued)*
Online Database Image Copy utility (DFSUICP0) 169
converting
DEDB to MSDB 350
MSDB to DEDB 350
DEDB
High-Speed Direct Reorganization utility 309
reorganizing
DEDBs 309
DATE= keyword
SB Test utility SELECT statement 437
DB0 statement
Database Change Accumulation utility (DFSUCUM0) 205
DB1 statement
Database Change Accumulation utility (DFSUCUM0) 206
DBDDDS= keyword
control statements
UCF FUNCTION=DR 382
UCF FUNCTION=DU 384
UCF FUNCTION=DX 386
UCF FUNCTION=IL 387
UCF FUNCTION=IM 388
UCF FUNCTION=PU 392
UCF FUNCTION=RU 395
UCF FUNCTION=SN 397
UCF FUNCTION=SR 399
UCF FUNCTION=SU 401
UCF FUNCTION=SX 403
UCF FUNCTION=ZB 405
DBDLIB
HALDB (High Availability Large Database) 114
member 92
DBDS select statement
concurrent copy function 151
fast replication function 153
DBFDBDR0 (MSDB Dump Recovery utility)
description 247
examples 252
input and output 249
JCL requirements
DD statements 250
EXEC statement 250
JCL specifications 250
prerequisites 248
recommendations 249
RECON status 248
requirements 248
restrictions 248
return codes 251
utility control statements 251
DBFDBMA0 (MSDB Maintenance utility)
description 117
examples 124
functions
deletion 126
insertion 126
modification 126
replacement 126
input and output 118

DBFDBMA0 (MSDB Maintenance utility) *(continued)*
JCL requirements
DD statements 120
EXEC statement 120
prerequisite 118
QUIESCE database status 118
recommendations 118
requirements 118
restrictions 118
return codes 120
utility control statements 121
action statement 121
run statement 121
DBFUCDB0 (MSDB-to-DEDB Conversion utility)
converting a DEDB to MSDB 350
DD statements 344
description 343
examples 347
EXEC statement 344
fallback 350
input and output 344
JCL specifications 344
prerequisites 343
QUIESCE database status 343
recommendations 343
requirements 343
restrictions 343
running the utility 350
summary report 344
utility control statements 346
DBFUHDR0 (High-Speed DEDB Direct Reorganization utility)
buffer usage 315
BUFNO command 315
control statements 313
DD statements 311
error processing 318
example JCL 312
EXEC statement 311
input and output 310
JCL specifications 311
overview 309
prerequisites 309
QUIESCE database status 309
recommendations 310
recovery and restart 317
requirements 310
restrictions 309
return codes 312
running the utility 315
segment shunting 316
statistics collected 310
DBFUMDL0 (DEDB Sequential Dependent Delete utility)
control statements 56
description 55
example JCL 60
input and output 56
JCL requirements
DD statements 56
EXEC statement 56
JCL specifications and examples 56
prerequisites 55
QUIESCE database status 55
range of processing 61
DBFUMDL0 (DEDB Sequential Dependent Delete utility) *(continued)*
recommendations 55
requirements 55
restrictions 55
return codes 56
DBFUMIN0 (DEDB Initialization utility)
description 49
examples 54
input and output 50
JCL requirements
DD statements 51
EXEC statement 51
prerequisites 49
QUIESCE database status 49
recommendations 50
requirements 49
restrictions 49
return codes 50
utility control statements 52
DBFUMMH0 (DEDB Area Data Set Compare utility)
control statements 232
description 229
examples 233
input and output 230
JCL requirements
DD statements 231
EXEC statement 231
JCL specifications 231
prerequisites 230
QUIESCE database status 229
recommendations 230
requirements 230
restrictions 229
return codes 231
DBFUMRI0 (DEDB Area Data Set Create utility)
control statements 238
description 235
examples 239
input and output 237
JCL requirements
DD statements 237
EXEC statement 237
JCL specifications 237
prerequisites 236
QUIESCE database status 236
recommendations 236
requirements 236
restrictions 236
return codes 237
DBFUMSC0 (DEDB Sequential Dependent Scan utility)
control statements 66
description 63
example JCL 71
input and output 64
JCL requirements
DD statements 65
EXEC statement 65
JCL specifications and examples 64
prerequisites 64
QUIESCE database status 64
range of scan 71
recommendations 64
requirements 64
DBFUMSC0 (DEDB Sequential Dependent Scan utility) *(continued)*
restrictions 64
return codes 64
DBIL= statement
Database Prereorganization utility (DFSURPR0) 262
DBIO statement
SB test utility (DFSSBHD0) 436
DBN= statement
MSDB Dump Recovery utility 252
MSDB Maintenance utility
action statement 122
MSDB Maintenance utility change statement 123
DBNAME= operand
control statements
UCF FUNCTION=DR 381
UCF FUNCTION=DU 383
UCF FUNCTION=DX 385
UCF FUNCTION=IL 386
UCF FUNCTION=IM 388
UCF FUNCTION=PR 390
UCF FUNCTION=PU 391
UCF FUNCTION=RR 392
UCF FUNCTION=RU 394
UCF FUNCTION=SN 397
UCF FUNCTION=SR 399
UCF FUNCTION=SU 401
UCF FUNCTION=SX 403
UCF FUNCTION=ZB 405
Database Surveyor utility (DFSPRSUR) 39
Partial DB Reorganization
Step 1 (DFSPRCT1) 360
Step 2 (DFSPRCT2) 361
DBR= statement
Database Prereorganization utility (DFSURPR0) 263
DBS= statement
Database Scan utility (DFSURGS0) 30
DD statements
Batch Backout utility (DFSBBO00) 185
Database Change Accumulation utility (DFSUCUM0) 199
Database Image Copy 2 utility (DFSUDMT0) 147
Database Image Copy utility (DFSUDMP0) 138
Database Prefix Resolution utility (DFSURG10) 9
Database Prefix Update utility (DFSURGP0) 16
Database Prereorganization utility (DFSURPR0) 260
Database Recovery utility (DFSURDB0) 218
Database Scan utility (DFSURGS0) 28
Database Surveyor utility (DFSPRSUR) 37
Database-Monitor Report Print utility (DFSUTR30) 422
DEDB Area Data Set Compare utility (DBFUMMH0) 231

DD statements *(continued)*
DEDB Area Data Set Create utility (DBFUMRI0) 237
DEDB Initialization utility (DBFUMIN0) 51
DEDB Sequential Dependent Delete utility (DBFUMDL0) 56
DEDB Sequential Dependent Scan utility (DBFUMSC0) 65
EXEC Statement 65
HD Reorganization Unload utility (DFSURGU0) 298
HISAM Reorganization Reload utility (DFSURRL0) 323
HISAM Reorganization Unload utility (DFSURUL0) 333
MSDB Dump Recovery utility (DBFDBDR0) 250
MSDB Maintenance utility (DBFDBMA0) 120
Online Database Image Copy utility (DFSUICP0) 172
Partial Database Reorganization utility (DFSPRCT1) 356
Partial Database Reorganization utility (DFSPRCT2) 356
Program-Isolation-Trace Report utility (DFSPIRP0) 427
SB Test utility (DFSSBHD0) 434
Utility Control Facility (DFSUCF00) 372
DDNAME
HALDB (High Availability Large Database) 90
DEDB
converting a DEDB to MSDB 350
converting an MSDB to DEDB 350
DEDB Area Data Set Compare utility (DBFUMMH0)
control statements 232
description 229
examples 233
input and output 230
JCL requirements
DD statements 231
EXEC statement 231
JCL specifications 231
prerequisites 230
QUIESCE database status 229
recommendations 230
requirements 230
restrictions 229
return codes 231
DEDB Area Data Set Create utility (DBFUMRI0)
control statements 238
description 235
examples 239
input and output 237
JCL requirements
DD statements 237
EXEC statement 237
JCL specifications 237
prerequisites 236
QUIESCE database status 236
recommendations 236
requirements 236
DEDB Area Data Set Create utility (DBFUMRI0) *(continued)*
restrictions 236
return codes 237
DEDB Initialization utility (DBFUMIN0)
description 49
examples 54
input and output 50
JCL requirements
DD statements 51
EXEC statement 51
prerequisites 49
QUIESCE database status 49
recommendations 50
requirements 49
restrictions 49
return codes 50
utility control statements 52
DEDB Sequential Dependent Delete utility (DBFUMDL0)
control statements 56
description 55
example JCL 60
input and output 56
JCL requirements
DD statements 56
EXEC statement 56
JCL specifications and examples 56
prerequisites 55
QUIESCE database status 55
range of processing 61
recommendations 55
requirements 55
restrictions 55
return codes 56
DEDB Sequential Dependent Scan utility (DBFUMSC0)
control statements 66
description 63
example JCL 71
EXEC Statement 65
input and output 64
JCL requirements
DD statements 65
EXEC statement 65
JCL specifications and examples 64
prerequisites 64
QUIESCE database status 64
range of scan 71
recommendations 64
requirements 64
restrictions 64
return codes 64
DFPXPMB 114, 115
DFSBBO00 (Batch Backout utility)
description 179
in an RSR environment 193
input and output 182
JCL requirements
DD statements 185
EXEC statement 185
prerequisites 181
QUIESCE database status 180
recommendations 181
Remote Site Recovery 193
requirements 181
restrictions 180
DFSBBO00 (Batch Backout utility) *(continued)*
return codes 183
utility control statements
ABEND statement 188
ABENDMSG statement 189
ACTIVE statement 189
BYPASS LOGVER statement 189
BYPASS SEQVER statement 190
CHKPT statement 190
COLDSTART statement 191
NOREADBACK statement 191
overview 188
READBACK statement 192
DFSMAID0 (HALDB Migration Aid utility)
data spaces 273
data spaces and analysis type 273
data spaces, calculating sizes 274
DD statements 269
example JCL 272
EXEC statement 269
input and output 268
JCL specifications 269
output messages and statistics 267
overview 267
prerequisites 268
QUIESCE database status 268
recommendations 268
requirements 268
restrictions 268
return codes 270
sampling 273, 274
utility control statements 271
DFSMSdss
SET PATCH commands 166
DFSPIRP0 (Program-Isolation-Trace Report utility)
description 425
example 429
input and output 426
JCL requirements
DD statements 427
EXEC statement 427
JOBLIB DD statement 427
JCL specifications and examples 427
prerequisites 425
recommendations 425
requirements 425
restrictions 425
return codes 428
utility control statements 428
DFSPRCT1
DFSPRCT2 (Partial Database Reorganization utility)
restricted from Utility Control Facility 369
Partial Database Reorganization utility 354
DFSPRCT1 (Partial Database Reorganization utility)
description 353
examples 362
prerequisites 354
QUIESCE database status 354
recommendations 354
requirements 354

DFSPRCT1 (Partial Database Reorganization utility) *(continued)*
restrictions 354
DFSPRCT1 and DFSPRCT2 (Partial Database Reorganization utility)
utility control statements 359
DFSPRCT2 (Partial Database Reorganization utility)
description 353
examples 362
prerequisites 354
QUIESCE database status 354
recommendations 354
requirements 354
restrictions 354
DFSPRCT2 (Unload/Reload/Pointer Resolution Step 2)
Partial Database Reorganization utility 355
DFSPREC0 (HALDB Index/ILDS Rebuild utility)
examples 245
free space option 246
input and output 242
JCL requirements
DD statements 243
EXEC statement 242
overview 241
QUIESCE database status 241
restrictions 241
return codes 244
utility control statements 244
VSAM load mode 246
DFSPRSUR (Database Surveyor utility)
description 35
examples 40
input 35
JCL requirements
DD statements 37
EXEC statement 36
output 36
prerequisites 35
QUIESCE database status 35
recommendations 35
requirements 35
restricted from Utility Control Facility 369
restrictions 35
return codes 36
utility control statements 38
DFSSBHD0 (SB Test utility)
description 431
example JCL 438
image capture log record 431
input 433
JCL example 438
JCL requirements
DD statements 434
EXEC statement 434
output 434
prerequisites 433
recommendations 433
requirements 433
restrictions 433
return codes 436
utility control statements
DBIO 436
DFSSBHD0 (SB Test utility) *(continued)*
utility control statements *(continued)*
overview 436
SELECT 437
DFSUCF00 (Utility Control Facility)
checkpoint/restart capabilities 414
database zap capability 416
description 369
error-point abends 416
execution of database utilities 410
FUNCTION= keyword
control statement requirements 378
initial load application program considerations
description 412
initial load exit routine 413
input and output 372
JCL requirements
DD statements 372
EXEC statement 372
module zap capability 416
prerequisites 370
QUIESCE database status 369
recommendations 372
restrictions 369, 370
service aids 416
termination/error processing 413
types of processing
normal 410
restart 414
user exit routine processing 415
utility control statements
FUNCTION=OP 380
overview 378
WTOR (write-to-operator-with-reply) function 417
DFSUCUM0 (Database Change Accumulation utility)
description 195
examples 208
input and output 197
JCL requirements
DD statements 199
EXEC statement 198
prerequisites 197
purge date and time 202
QUIESCE database status 195
recommendations 197
requirements 197
restricted from Utility Control Facility 369
restrictions 196
return codes 201
utility control statements
DB0 statement 205
DB1 statement 206
ID statement 204
overview 201
SO statement 207
DFSUDMP0 (Database Image Copy utility)
description 131
examples 140
in an RSR environment 134
input and output 135
DFSUDMP0 (Database Image Copy utility) *(continued)*
JCL requirements
DD statements 138
EXEC statement 135
JCL specifications 135
prerequisites 133
QUIESCE database status 132
Remote Site Recovery 134
requirements 133
restrictions 132
return codes 135, 146
utility control statements 140
DFSUDMT0 (Database Image Copy 2 utility)
concurrent copy function 160
copy complete notification 164
data sets, multiple input 161
description 143
DFSMSdss SET PATCH commands 166
dynamic data set names 164
examples 150
fast replication function 161
group names 162
input and output 145
JCL requirements
DD statements 147
EXEC statement 146
prerequisites 144
QUIESCE database status 143
recommendations 145
requirements 144
restrictions 143
same data set option 163
utility control statements
concurrent copy DBDS select statement 151
fast replication DBDS select statement 153
group name statement 155
HLQ specification statement 157
overview 151
SET PATCH control statement 159
DFSUICP0 (Online Database Image Copy utility)
description 169
example 175
JCL requirements
DD statements 172
EXEC statement 172
output 171
prerequisites 170
QUIESCE database status 169
recommendations 170
recovery and restart 176
requirements 170
restricted from Utility Control Facility 369
restrictions 169
return codes 171
utility control statements 174
DFSUPNT0 (HALDB Partition Data Set Initialization utility)
DD statements 78
examples 80

DFSUPNT0 (HALDB Partition Data Set Initialization utility) *(continued)*
EXEC statement 77
input and output 76
JCL specifications and examples 77
overview 75
prerequisites 76
QUIESCE database status 76
recommendations 76
requirements 76
restrictions 76
return codes 76
utility control statements 79
DFSURDB0 (Database Recovery utility)
description 211
DFSUSNAP DD statement 219
examples 224
input and output 215
JCL requirements
DD statements 218
EXEC statement 218
prerequisites 213
QUIESCE database status 213
recommendations 215
Remote Site Recovery 211
requirements 213
restrictions 213
return codes 220
RSR environment 211
SNAP output data set attributes 219
utility control statements
ABEND 221
NOSEQCK 221
overview 221
S (database recovery) 221
DFSURG10 (Database Prefix Resolution utility)
example JCL 11
EXEC statement example 7
EXEC statement PARM field options 7
input and output 6
introduction 3
JCL example 11
JCL requirements
DD statements 9
EXEC statement 7
prerequisites 5
QUIESCE database status 4
recommendations 6
requirements 5
restrictions 4
return codes 6
z/OS SORT/MERGE program 7
DFSURGL0 (HD Reorganization Reload utility)
description 275
examples 286
free space option 278
output messages and statistics 280
prerequisites 277
QUIESCE database status 277
recommendations 278
reports 284
requirements 277
restrictions 277
return codes 283
DFSURGL0 (HD Reorganization Reload utility) *(continued)*
statistics 284
utility control statements 283
VSAM load mode 278
DFSURGP0 (Database Prefix Update utility)
input and output 15
introduction 13
JCL requirements
DD statements 16
EXEC statement 16
prerequisites 15
QUIESCE database status 14
recommendations 15
requirements 15
restart 24
restrictions 14
utility control statements
ABEND 19
CHKPT 18
overview 18
RSTRT 18
SNAP 19
DFSURGS0 (Database Scan utility)
description 25
example JCL 32
input and output 27
JCL requirements
DD statements 28
EXEC statement 28
JCL specifications and examples 27
output messages 27
prerequisites 27
QUIESCE database status 27
recommendations 27
requirements 27
restart 33
restrictions 27
return codes 27
scan options
SEG 30
SEQ 30
utility control statements
ABEND 32
CHKPT 31
DBS 30
overview 30
RSTRT 31
DFSURGU0 (HD Reorganization Unload utility)
DD statements 298
description 291
example JCL 302
EXEC statement 298
input and output 296
JCL requirements
DD statements 298
JCL specifications 298
prerequisites 294
QUIESCE database status 293
recommendations 296
RECON status 293
requirements 294
restrictions 293
return codes 300
utility control statements 301
DFSURPR0 (Database Prereorganization utility)
description 257
example 264
JCL requirements
DD statements 260
EXEC statement 260
output messages 259
QUIESCE database status 258
restrictions 258
return codes 262
utility control statements
DBIL= 262
DBR= 263
OPTIONS= 263
overview 262
DFSURRL0 (HISAM Reorganization Reload utility)
description 319
example JCL 326
input and output 320
JCL requirements
DD statements 323
EXEC statement 323
JCL specifications 323
output messages and statistics 320
prerequisites 320
QUIESCE database status 320
recommendations 320
requirements 320
restrictions 320
return codes 324
utility control statements 325
DFSURUL0 (HISAM Reorganization Unload utility)
CHANGE= statement 337
description 327
examples 338
input and output 329
JCL requirements
DD statements 333
EXEC statement 333
JCL specifications 333
OPTIONS= statement 338
output messages and statistics 330
prerequisites 329
QUIESCE database status 328
recommendations 329
requirements 329
restrictions 328
return codes 335
utility control statements
overview 336
DFSUSNAP DD statement
attributes 219
DFSUTR30 (Database-Monitor Report Print utility)
analysis control data set 422
description 421
example 423
input and output 422
JCL requirements
DD statements 422
EXEC statement 422
JCL specifications 422
prerequisites 421
recommendations 421

DFSUTR30 (Database-Monitor Report Print utility) *(continued)*
requirements 421
restrictions 421
return codes 423
DSDDNAM= data set
UCF FUNCTION=IL control statement 387
DSPXPDA 111
DSPXPEA 112
database name 112
output data set name 112
DSPXPIA 112
database name 113
input data set name 113
input member name 113
processing option 113
DSPXPKE panel 89
DSPXPLA 104
act 104
data set name prefix 104
ID 104
name 104
DSPXPLB 105
DSPXPOA 111
DSPXRUN command 87

E

ESCISZ= keyword
HISAM Reorganization Unload utility (DFSURUL0) 337
ESREC= keyword
HISAM Reorganization Unload utility (DFSURUL0) 337
EXEC statement 65
Batch Backout utility (DFSBBO00) 185
Database Change Accumulation utility (DFSUCUM0) 198
Database Image Copy 2 utility (DFSUDMP0) 146
Database Image Copy utility (DFSUDMP0) 135
Database Prefix Resolution utility (DFSURG10) 7
Database Prefix Update utility (DFSURGP0) 16
Database Prereorganization utility (DFSURPR0) 260
Database Recovery utility (DFSURDB0) 218
Database Scan utility (DFSURGS0) 28
Database Surveyor utility (DFSPRSUR) 36
Database-Monitor Report Print utility (DFSUTR30) 422
DBFUCDB0 (MSDB-to-DEDB Conversion utility) 344
DBFUHDR0 (High-Speed DEDB Direct Reorganization utility) 311
DEDB Area Data Set Compare utility (DBFUMMH0) 231
DEDB Area Data Set Create utility (DBFUMRI0) 237
EXEC statement *(continued)*
DEDB Initialization utility (DBFUMIN0) 51
DEDB Sequential Dependent Delete utility (DBFUMDL0) 56
HD Reorganization Unload utility (DFSURGU0) 298
High-Speed DEDB Direct Reorganization utility (DBFUHDR0) 311
HISAM Reorganization Reload utility (DFSURRL0) 323
HISAM Reorganization Unload utility (DFSURUL0) 333
MSDB Dump Recovery utility (DBFDBDR0) 250
MSDB Maintenance utility (DBFDBMA0) 120
MSDB-to-DEDB Conversion utility (DBFUCDB0) 344
Online Database Image Copy utility (DFSUICP0) 172
Partial Database Reorganization utility (DFSPRCT1) 355
Partial Database Reorganization utility (DFSPRCT2) 356
Program-Isolation-Trace Report utility (DFSPIRP0) 427
SB Test utility (DFSSBHD0) 434
Utility Control Facility (DFSUCF00) 372
EXEC= statement
control statements
UCF FUNCTION=DR 382
UCF FUNCTION=DU 383
UCF FUNCTION=DX 385
UCF FUNCTION=IL 387
UCF FUNCTION=IM 389
UCF FUNCTION=PR 390
UCF FUNCTION=PU 391
UCF FUNCTION=RR 393
UCF FUNCTION=RU 395
UCF FUNCTION=SN 397
UCF FUNCTION=SR 400
UCF FUNCTION=SU 401
UCF FUNCTION=SX 403
UCF FUNCTION=ZB 406
EXITRLD= keyword
control statements
UCF FUNCTION=DX 385
UCF FUNCTION=SX 404
EXITRTN= keyword
control statements
UCF FUNCTION=DR 382
UCF FUNCTION=DU 384
UCF FUNCTION=IL 387
UCF FUNCTION=IM 389
UCF FUNCTION=PR 390
UCF FUNCTION=PU 391
UCF FUNCTION=RR 393
UCF FUNCTION=RU 395
UCF FUNCTION=SN 398
UCF FUNCTION=SR 400
UCF FUNCTION=SU 401
UCF FUNCTION=SX 404
UCF FUNCTION=ZB 406
UCF FUNCTION=ZM 408
EXTRACT= keyword
UCF FUNCTION=RU control statement 396

F

Fast Path
DEDB Sequential Dependent Delete utility (DBFUMDL0) 55
DEDB utilities
command continuation 52, 53, 57, 66, 232, 238, 313
DEDB Area Data Set Compare utility (DBFUMMH0) 229
DEDB Area Data Set Create utility (DBFUMRI0) 235
High-Speed Direct Reorganization utility 309
Initialization utility (DBFUMIN0) 49
Sequential Dependent Scan utility (DBFUMSC0) 63
DEDB, reorganization
High-Speed Direct Reorganization utility 309
MSDB utilities
MSDB Dump Recovery utility (DBFDBDR0) 247
MSDB Maintenance utility (DBFDBMA0) 117
MSDBINIT DATASET 123
MSDB-to-DEDB Conversion utility (DBFUCDB0) 343
Fast Path databases 3
fast replication
Database Image Copy 2 utility (DFSUDMT0)
dynamic data set names 164
fast replication function
DBDS select statement 153
FIELD= keyword
MSDB Maintenance utility MSDBINIT statement 124
File Action bar 91
FIXED= keyword
MSDB Maintenance utility action statement 122
free space
ILDS
including free space during rebuild 246, 278
FROMAREA= operand
DB Database Surveyor utility 39
Partial DB Reorganization
Step 1 (DFSPRCT1) 360
FUNCTION= keyword
Utility Control Facility (UCF)
DR for HD RR Reload utility 381
DU for HD Reorganization Unload utility 382
DX for HD RR Unload and Reload utilities (combined) 384
IL for initial load program 386
IM for Image Copy utility 388
PR for Prefix Resolution utility 389
PU for Prefix Update utility 390

FUNCTION= keyword *(continued)*
Utility Control Facility (UCF) *(continued)*
RR for Secondary Index Reload 392
RU for Secondary Index Unload 393
SN for Database Scan utility 396
SR for HISAM Reorganization Reload utility 398
SU for HISAM Reorganization Unload utility 400
SX for HISAM Reorganization Unload and Reload utilities (combined) 402
ZB for database zaps 404
ZM for module zaps 406
FUNCTION=OP statement
Utility Control Facility (DFSUCF00) 380

G

group name statement
utility control statements, DFSUDMT0 155

H

HALDB (High Availability Large Database)
automatic partition definition using Partition Definition utility 97
batch import 85
Change Partition screen 99
F11 101
F5 101
F6 101
Partition high key 100
Partition ID field 99
Partition Selection String 100
configuration
list 115
Database Partition list
act 106
anchor 106
bytes 106
FBFF 106
File action bar choice 104
FSFF 106
high block 106
module 106
name 106
Database Partitions list 103
displaying 103
DBDLIB 114
defining data set groups using Partition Definition utility 101
Delete a Database panel using Partition Definition utility 111
DFSMAID0 (HALDB Migration Aid utility) 267
DSPXPDA 111
DSPXPEA
See DSPXPEA 112
DSPXPGA 102
HALDB (High Availability Large Database) *(continued)*
DSPXPGB 102
DSPXPGD 103
DSPXPIA
See DSPXPIA 112
DSPXPKE panel 89
DSPXPLA
See DSPXPLA 103
DSPXPLB 105
DSPXPOA 111
DSPXRUN command 87
File Action bar 91
actions 91
HALDB Index/ILDS Rebuild utility 241
HALDB Migration Aid utility (DFSMAID0) 267
HALDB Partition Definition utility (%DFSHALDB)
main panel 89
main screen 89
Partitioned Databases panel 89
using 89
ILDS
rebuilding, DFSPREC0 241
importing database information using Partition Definition utility 112
IMS configuration 114
information
changing using Partition Definition utility 91
deleting using Partition Definition utility 91, 111
exporting using Partition Definition utility 92, 112
importing using Partition Definition utility 92, 112
opening using Partition Definition utility 111
viewing using Partition Definition utility 91
interfaces
HALDB Partition Definition utility (%DFSHALDB) 89
ISRDDN 113
line commands 106
manual partition definition using Partition Definition utility 98
master
values 92
Partition Definition utility 98
accessing 89
displaying HALDB data set concatenation 113
high key value 98
impact on RECON 81
modifying fields 98
panels 89
partitions
changing using Partition Definition utility 91
copying using Partition Definition utility 107
creating using Partition Definition utility 91
HALDB (High Availability Large Database) *(continued)*
partitions *(continued)*
defining using Partition Definition utility 92
deleting using Partition Definition utility 106
manual definition using Partition Definition utility 98
maximum number 92
opening using Partition Definition utility 92, 107
printing information using Partition Definition utility 107
performance
ILDS 246, 279
RECON data set 114
HALDB (High Availability Large Database) partition definition utility
registering OLR capability with DBRC 93
HALDB Index/ILDS Rebuild utility (DFSPREC0)
examples 245
free space option 246
input and output 242
JCL requirements
DD statements 243
EXEC statement 242
overview 241
QUIESCE database status 241
restrictions 241
return codes 244
utility control statements 244
VSAM load mode 246
HALDB Migration Aid utility (DFSMAID0)
data spaces 273
data spaces and analysis type 273
data spaces, calculating sizes 274
DD statements 269
example JCL 272
EXEC statement 269
input ant output 268
JCL specifications 269
output messages and statistics 267
overview 267
prerequisites 268
QUIESCE database status 268
recommendations 268
requirements 268
restrictions 268
return codes 270
sampling 273, 274
utility control statements 271
HALDB Partition Data Set Initialization utility (DFSUPNT0)
DD statements 78
examples 80
EXEC statement 77
input and output 76
JCL specifications and examples 77
overview 75
prerequisites 76
QUIESCE database status 76
recommendations 76
requirements 76

HALDB Partition Data Set Initialization utility (DFSUPNT0) *(continued)*
restrictions 76
return codes 76
utility control statements 79
HALDB Partition Definition utility (%DFSHALDB)
batch JCL requirements
DD statements 85
examples 86, 88
export 89
foreground JCL requirements
DD statements 84
import 89
input and output 82
interactive application 89
ISPF panels
HALDB Partition Definition utility (%DFSHALDB) 89
main panel 89
main screen 89
overview 81
Partitioned Databases panel 89
prerequisites 82
recommendations 82
requirements 82
restrictions 81
return codes 86
using 89
utility control statements 87
HD Reorganization Reload utility (DFSURGL0)
DD statements 280
description 275
examples 286
EXEC statement 280
free space option 278
JCL 280
output messages and statistics 280
prerequisites 277
QUIESCE database status 277
recommendations 278
reports 284
requirements 277
restrictions 277
return codes 283
statistics 284
utility control statements 283
VSAM load mode 278
HD Reorganization Unload utility (DFSURGU0)
DD statements 298
description 291
example JCL 302
EXEC statement 298
input and output 296
JCL requirements
DD statements 298
JCL specifications 298
prerequisites 294
QUIESCE database status 293
recommendations 296
RECON status 293
requirements 294
restrictions 293
return codes 300
utility control statements 301
High Availability Large Database (HALDB)
DFSMAID0 (HALDB Migration Aid utility) 267
HALDB Index/ILDS Rebuild utility 241
HALDB Migration Aid utility (DFSMAID0) 267
ILDS
rebuilding, DFSPREC0 241
performance
ILDS 246, 279
High-Speed DEDB Direct Reorganization utility (DBFUHDR0)
buffer usage 315
BUFNO command 315
control statements 313
DD statements 311
error processing 318
example JCL 312
EXEC statement 311
input and output 310
JCL specifications 311
overview 309
prerequisites 309
QUIESCE database status 309
recommendations 310
recovery and restart 317
requirements 310
restrictions 309
return codes 312
running the utility 315
segment shunting 316
statistics collected 310
HISAM Reorganization Reload utility (DFSURRL0)
description 319
example JCL 326
input and output 320
JCL requirements
DD statements 323
EXEC statement 323
JCL specifications 323
output messages and statistics 320
prerequisites 320
QUIESCE database status 320
recommendations 320
requirements 320
restrictions 320
return codes 324
utility control statements 325
HISAM Reorganization Unload utility (DFSURUL0)
CHANGE= statement 337
description 327
examples 338
input and output 329
JCL requirements
DD statements 333
EXEC statement 333
JCL specifications 333
OPTIONS= statement 338
output messages and statistics 330
prerequisites 329
QUIESCE database status 328
recommendations 329
requirements 329
HISAM Reorganization Unload utility (DFSURUL0) *(continued)*
restrictions 328
return codes 335
utility control statements
overview 336
HLQ specification statement
utility control statements, DFSUDMT0 157

I

ID statement
Database Change Accumulation utility (DFSUCUM0) 204
IDXIN= keyword
UCF FUNCTION=RU control statement 396
ILDS (indirect list data set)
free space
including free space during rebuild 246, 278
performance
free space, rebuilding with 246, 279
rebuilding
free space option 246, 278
VSAM load mode 246, 278
ILPGM= parameter
UCF FUNCTION=IL control statement 386
ILPSBNAM= keyword
UCF FUNCTION=IL control statement 387
image copies
Database Image Copy 2 utility (DFSUDMT0) 143
copy complete notification 164
Database Image Copy utility (DFSUDMP0) 131
Online Database Image Copy utility (DFSUICP0) 169
INCR= keyword
MSDB Maintenance utility action statement 122
INDDS= keyword
control statements
UCF FUNCTION=DR 381
UCF FUNCTION=DX 385
UCF FUNCTION=PR 390
UCF FUNCTION=PU 392
UCF FUNCTION=RR 393
UCF FUNCTION=SR 399
UCF FUNCTION=SX 403
indirect list data set (ILDS)
free space
including free space during rebuild 246, 278
performance
free space, rebuilding with 246, 279
rebuilding
free space option 246, 278
VSAM load mode 246, 278
ISPF
batch job 85
ISPF member list 90

ISPF member list *(continued)*
display 90
ISPSTART 87
ISPTLIB 112

J

JOB= keyword
SB Test utility SELECT statement 437
JOBLIB DD statement
Program-Isolation-Trace Report utility (DFSPIRP0) 427

K

KDSDD= keyword
control statements
UCF FUNCTION=RR 393
UCF FUNCTION=RU 394
UCF FUNCTION=SR 399
UCF FUNCTION=SU 399
UCF FUNCTION=SX 403
KEY= operand
MSDB Maintenance utility MSDBINIT statement 123
keyboard shortcuts ix
KEYRANGE= operand
DB Database Surveyor utility 39
Partial DB Reorganization
Step 1 (DFSPRCT1) 360
KSCISZ= keyword
HISAM Reorganization Unload utility (DFSURUL0) 337
KSREC= keyword
HISAM Reorganization Unload utility (DFSURUL0) 337

L

legal notices
notices 441
trademarks 441

M

MODE= keyword
Database Surveyor utility (DFSPRSUR) 40
MSDB Maintenance utility
action statement 122
module zap capability
Utility Control Facility (DFSUCF00) 416
MSDB
converting a DEDB to MSDB 350
converting an MSDB to DEDB 350
MSDB database
deleting 126
inserting 126
modifying 126
replacing 126
MSDB Dump Recovery utility (DBFDBDR0)
description 247
examples 252
MSDB Dump Recovery utility (DBFDBDR0) *(continued)*
input and output 249
JCL requirements
DD statements 250
EXEC statement 250
JCL specifications 250
prerequisites 248
recommendations 249
RECON status 248
requirements 248
restrictions 248
return codes 251
utility control statements 251
MSDB Maintenance utility (DBFDBMA0)
description 117
examples 124
functions
deletion 126
insertion 126
modification 126
replacement 126
input and output 118
JCL requirements
DD statements 120
EXEC statement 120
prerequisite 118
QUIESCE database status 118
recommendations 118
requirements 118
restrictions 118
return codes 120
utility control statements 121
action statement 121
run statement 121
MSDB-to-DEDB Conversion utility
DBT Unload/Reload, using a tool other than 351
fallback 350
using a tool other than DBT Unload/Reload 351
MSDB-to-DEDB Conversion utility (DBFUCDB0)
converting a DEDB to MSDB 350
DD statements 344
description 343
examples 347
EXEC statement 344
input and output 344
JCL specifications 344
prerequisites 343
QUIESCE database status 343
recommendations 343
requirements 343
restrictions 343
running the utility 350
summary report 344
utility control statements 346
MSDBDUMP data set
/DBDUMP command 249
MSDB Dump Recovery utility 247
MSGNUM= keyword
UCF FUNCTION=OP control statement 381

N

NBR= keyword
SB Test utility SELECT statement 438
NOPUNCH parameter
Database Prereorganization utility (DFSURPR0) 264
NOREADBACK statement
Batch Backout utility (DFSBBO00) 191
NOSEQCK statement
Database Recovery utility (DFSURDB0) 221
NSTAT keyword
HISAM Reorganization Reload utility (DFSURRL0) 325
HISAM Reorganization Unload utility (DFSURUL0) 338

O

OLR (online reorganization)
HALDB (High Availability Large Database)
registering OLR capability with DBRC using PDU 93
Online Database Image Copy utility (DFSUICP0)
description 169
example 175
JCL requirements
DD statements 172
EXEC statement 172
output 171
prerequisites 170
QUIESCE database status 169
recommendations 170
recovery and restart 176
requirements 170
restricted from Utility Control Facility 369
restrictions 169
return codes 171
utility control statements 174
online reorganization (OLR)
HALDB (High Availability Large Database)
registering OLR capability with DBRC using PDU 93
OPTIONS= statement
Database Prereorganization utility (DFSURPR0) 263
HISAM Reorganization Reload utility (DFSURRL0) 325
HISAM Reorganization Unload utility (DFSURUL0) 338
OUTDDS= parameter
control statements
UCF FUNCTION=DU 383
UCF FUNCTION=DX 385
UCF FUNCTION=IL 387
UCF FUNCTION=IM 388
UCF FUNCTION=PR 390
UCF FUNCTION=RU 394
UCF FUNCTION=SN 397
UCF FUNCTION=SU 401
UCF FUNCTION=SX 403

P

Partial Database Reorganization utility (DFSPRCT2)
  step 2 utility control statement 359
Partial Database Reorganization utility (DFSPRCT1 and DFSPRCT2)
  checkpoint/restart 353
  DD statements (Step 2) 356
  description 353
  examples 362
  JCL requirements
    DD statements (Step 1) 356
    EXEC statement (Step 1) 355
    EXEC statement (Step 2) 356
  prerequisites 354
  QUIESCE database status 354
  recommendations 354
  requirements 354
  restricted from Utility Control Facility 369
  restrictions 354
  return codes 359
  step 1 prereorganization 354
  step 2 unload/reload pointer resolution 355
  utility control statements 359
Partial Database Reorganization utility (DFSPRCT1)
  step 1 utility control statement 359
Partition Default information screen
  anchor 95
  automatic definition 94, 97
  block size 96
  bytes 95
  data set name prefix 95
  database name 94
  default JCL 96
  free block freq. factor 95
  free space percentage 95
  high block number 95
  image copy JCL 96
  input data set 94
  max. image copies 96
  module name 95
  online image copy JCL 96
  partition ID 95
  receive JCL 96
  recovery period 96
  recovery utility JCL 96
  reusable? 97
  use defaults for DS groups 94
partition definition utility
  HALDB (High Availability Large Database)
    registering OLR capability with DBRC 93
partitioned database
  information screen
    database name 92
    database organization 93
    number of data set groups 93
    part. selection routine 92
    recoverable? 93
    RSR global service group 92
    RSR tracking type 93
    share level 93
PDS directory 91
PROC= keyword
  MSDB Maintenance utility (DBFDBMA0) 121
Program-Isolation-Trace Report utility (DFSPIRP0)
  description 425
  example 429
  input and output 426
  JCL requirements
    DD statements 427
    EXEC statement 427
    JOBLIB DD statement 427
  JCL specifications and examples 427
  prerequisites 425
  recommendations 425
  requirements 425
  restrictions 425
  return codes 428
  utility control statements 428
PSB= operand
  Partial DB Reorganization Step 1 (DFSPRCT1) 361
publications
  changes xi
  organization xi
  titles xi
PUNCH parameter
  Database Prereorganization utility (DFSURPR0) 264

R

RBNID=keyword
  UCF FUNCTION=ZB control statement 406
READBACK statement
  Batch Backout utility (DFSBBO00) 192
RECON data set
  HALDB (High Availability Large Database) 114
RECOVCP parameter
  MSDB Dump Recovery 251
recover databases 177
recovery
  MSDB Dump Recovery 251
RELATE= keyword
  UCF FUNCTION=ZB control statement 406
  UCF FUNCTION=ZM control statement 408
Reload utility (DFSURGL0)
  reports 284
  statistics 284
Remote Site Recovery (RSR)
  DBTRACK 93
  global service group
    HALDB (High Availability Large Database) 92
  RCVTRACK 93
  shadowing
    HALDB (High Availability Large Database) 93
  tracking type
    HALDB (High Availability Large Database) 93
reorganization
  DEDB
    segment shunting 316
reorganize databases 255
REP= keyword
  UCF FUNCTION=ZB control statement 405
  UCF FUNCTION=ZM control statement 408
report and test databases 419
REQUEST= keyword
  control statements
    UCF FUNCTION=DU 383
    UCF FUNCTION=IM 388
    UCF FUNCTION=OP 380
    UCF FUNCTION=RR 393
    UCF FUNCTION=RU 395
    UCF FUNCTION=SN 397
    UCF FUNCTION=SR 399
    UCF FUNCTION=SU 401
    UCF FUNCTION=SX 403
RESTART= keyword
  Partial Database Reorganization Step 2 (DFSPRCT2) 361
RSTRT control statement
  Database Prefix Update utility (DFSURGP0) 18
RSTRT= statement
  Database Scan utility (DFSURGS0) 31
run statement
  MSDB Maintenance utility (DBFDBMA0) 121

S

S (database recovery) statement
  Database Recovery utility (DFSURDB0) 221
sampling
  HALDB Migration Aid utility (DFSMAID0) 274
SB Test utility (DFSSBHD0)
  description 431
  example JCL 438
  image capture log record 431
  input 433
  JCL example 438
  JCL requirements
    DD statements 434
    EXEC statement 434
  output 434
  prerequisites 433
  recommendations 433
  requirements 433
  restrictions 433
  return codes 436
  utility control statements
    DBIO 436
    overview 436
    SELECT 437
SCANSEG= keyword
  Partial Database Reorganization Step 2 (DFSPRCT2) 361
SEG option
  Database Scan utility (DFSURGS0) 30

segment shunting 316
segments
  segment shunting, DEDB 316
SEGNAME= keyword
  UCF FUNCTION=SN control statement 397
SELECT statement
  SB test utility (DFSSBHD0) 437
SEQ option
  Database Scan utility (DFSURGS0) 30
SEQ= keyword
  control statements
    UCF FUNCTION=DR 381
    UCF FUNCTION=DX 385
    UCF FUNCTION=IL 387
    UCF FUNCTION=IM 388
    UCF FUNCTION=PR 390
    UCF FUNCTION=PU 391
    UCF FUNCTION=RR 393
    UCF FUNCTION=RU 395
    UCF FUNCTION=SN 397
    UCF FUNCTION=SR 399
    UCF FUNCTION=SU 401
    UCF FUNCTION=SX 403
    UCF FUNCTION=ZM 408
sequential buffering
  problem determination
    SB Test utility (DFSSBHD0) 431
  SB Test utility (DFSSBHD0) 431
  testing
    SB Test utility (DFSSBHD0) 431
  tuning
    SB Test utility (DFSSBHD0) 431
SET PATCH
  DFSMSdss commands 166
SET PATCH control statement
  utility control statements, DFSUDMT0 159
SICON= keyword
  UCF FUNCTION=RU control statement 395
SNAP control statement
  Database Prefix Update utility (DFSURGP0) 19
SNAP output data set DCB attributes 219
SO statement
  Database Change Accumulation utility (DFSUCUM0) 207
SORTOPT= keyword
  Partial DB Reorganization Step 1 (DFSPRCT1) 361
STARTHEXT command
  DEDB online utilities 68
STAT parameter
  Database Prereorganization utility (DFSURPR0) 264
STATS keyword
  HISAM Reorganization Reload utility (DFSURRL0) 325
STATS parameter
  HISAM Reorganization Unload utility (DFSURUL0) 338
STOPHEXT command
  DEDB online utilities 58, 69
SUMM parameter
  Database Prereorganization utility (DFSURPR0) 264
syntax diagram
  how to read viii

T

TIME= keyword
  SB Test utility SELECT statement 437
TO= keyword
  MSDB Maintenance utility MSDBINIT statement 123
TOAREA= keyword
  Database Surveyor utility (DFSPRSUR) 40
  Partial DB Reorganization Step 1 (DFSPRCT1) 360
TRACE= keyword
  MSDB Maintenance utility action statement 122
trademarks 441

U

UNLOAD parameter
  MSDB Dump Recovery utility 251
UNLOADCP parameter
  MSDB Dump Recovery utility 251
user exit routine processing 415
utilities
  Batch Backout utility (DFSBBO00) 179
  Database Change Accumulation utility (DFSUCUM0) 195
  Database Image Copy 2 utility (DFSUDMT0) 143
  Database Image Copy utility (DFSUDMP0) 131
  Database Prefix Resolution utility (DFSURG10) 3
  Database Prefix Update utility (DFSURGP0) 13
  Database Prereorganization utility (DFSURPR0) 257
  Database Recovery utility (DFSURDB0) 211
  Database Scan utility (DFSURGS0) 25
  Database Surveyor utility (DFSPRSUR) 35
  Database-Monitor Report Print utility (DFSUTR30) 421
  DBFDBMA0 117
  DEDB Area Data Set Compare utility (DBFUMMH0) 229
  DEDB Area Data Set Create utility (DBFUMRI0) 235
  DEDB Initialization utility (DBFUMIN0) 49
  DEDB Sequential Dependent Delete utility (DBFUMDL0) 55
  DEDB Sequential Dependent Scan utility (DBFUMSC0) 63
  DFSPRCT1 353
  DFSPRCT2 353
utilities *(continued)*
  DFSPRSUR 35
  DFSUCF00 369
  DFSURG10 3
  DFSURGL0 275
  DFSURGP0 13
  DFSURGS0 25
  DFSURGU0 291
  DFSURPR0 257
  DFSURRL0 319
  DFSURUL0 327
  HD Reorganization Reload utility (DFSURGL0) 275
  HD Reorganization Unload utility (DFSURGU0) 291
  High-Speed DEDB Direct Reorganization utility (DBFUHDR0) 309
  HISAM Reorganization Reload utility (DFSURRL0) 319
  HISAM Reorganization Unload utility (DFSURUL0) 327
  MSDB Dump Recovery utility (DBFDBDR0) 247
  MSDB Maintenance utility (DBFDBMA0) 117
  Online Database Image Copy utility (DFSUICP0) 169
  Partial Database Reorganization utility (DFSPRCT1 and DFSPRCT2) 353
  Program-Isolation-Trace Report utility (DFSPIRP0) 425
  SB Test utility (DFSSBHD0) 431
  Utility control facility (DFSUCF00) 369
Utility Control Facility (DFSUCF00)
  checkpoint/restart capabilities 414
  database zap capability 416
  database zaps performed 403
  description 369
  error-point abends 416
  examples, JCL 409
  execution of database utilities 410
  FUNCTION= keyword
    control statement requirements 378
    DR for HD Reorganization Reload utility 381
    DU for HD Reorganization Unload utility 382
    IL for initial load program 386
    IM for Image Copy utility 388
    PR for Prefix Resolution utility 389
    PU for Prefix Update utility 390
    RR for Secondary Index Reload 392
    RU for Secondary Index Unload 393
    SN for Database Scan utility 396
    SR for HISAM Reorganization Reload utility 398
    SU for HISAM Reorganization Unload utility 400
    SX for HISAM Reorganization Unload and Reload 402
    ZB for database zaps 404

Utility Control Facility (DFSUCF00) *(continued)*
FUNCTION= keyword *(continued)*
ZM for module zaps 406
initial load application program considerations
checkpoint module (DFSUCP90), UCF 413
description 412
DL/I status codes associated 412
initial load exit routine 413
input and output 372
JCL requirements
DD statements 372
EXEC statement 372
keywords specified on utility control statements
minimum requirements 377
module zap capability 416
prerequisites 370
QUIESCE database status 369
recommendations 372
restrictions 369, 370
return codes 377
service aids 416
termination/error processing 413
types of processing
normal 410
restart 414
termination/error 413
user exit routine processing 415
utility control statements
Database Scan (SN) 396
database zaps (ZB) 404
FUNCTION=OP 380
HD Reorganization Reload (DR) 381
HD Reorganization Unload (DU) 382
HD Reorganization Unload and Reload combined (DX) 384
HISAM Reorganization Reload (SR) 398
HISAM Reorganization Unload (SU) 400
HISAM Reorganization Unload and Reload combined (SX) 402
Image Copy (IM) 388
initial load (IL) 386
module zaps (ZM) 406
overview 378
Prefix Resolution (PR) 389
Prefix Update (PU) 390
Secondary Index Reload (RR) 392
Secondary Index Unload (RU) 393
user's initial load (IL) 386
WTOR (write-to-operator-with-reply) function 417
utility control statements
%DFSHALDB (HALDB Partition Definition utility) 87
Batch Backout utility (DFSBBO00)
overview 188
Database Change Accumulation utility (DFSUCUM0)
overview 201
utility control statements *(continued)*
Database Image Copy 2 utility (DFSUDMT0)
overview 151
Database Image Copy utility (DFSUDMP0) 140
Database Prefix Update utility (DFSURGP0)
overview 18
Database Prereorganization utility (DFSURPR0)
overview 262
Database Recovery utility (DFSURDB0)
overview 221
Database Scan utility (DFSURGS0)
overview 30
Database Surveyor utility (DFSPRSUR) 38
DBFDBDR0 (MSDB Dump Recovery utility) 251
DBFDBMA0 (MSDB Maintenance utility) 121
DBFUCDB0 (MSDB-to-DEDB Conversion utility) 346
DBFUMIN0 (DEDB Initialization utility) 52
DEDB Initialization utility (DBFUMIN0) 52
DFSBBO00 (Batch Backout utility)
overview 188
DFSMAID0 (HALDB Migration Aid utility) 271
DFSPIRP0 (Program-Isolation-Trace Report utility) 428
DFSPRCT1 and DFSPRCT2 (Partial Database Reorganization utility) 359
DFSPREC0 (HALDB Index/ILDS Rebuild utility) 244
DFSPRSUR (Database Surveyor utility) 38
DFSSBHD0 (SB Test utility)
overview 436
DFSUCF00 (Utility Control Facility)
overview 378
DFSUCUM0 (Database Change Accumulation utility)
overview 201
DFSUDMP0 (Database Image Copy utility) 140
DFSUDMT0 (Database Image Copy 2 utility)
overview 151
DFSUICP0 (Online Database Image Copy utility) 174
DFSUPNT0 (HALDB Partition Data Set Initialization utility) 79
DFSURDB0 (Database Recovery utility)
overview 221
DFSURGL0 (HD Reorganization Reload utility) 283
DFSURGP0 (Database Prefix Update utility)
overview 18
utility control statements *(continued)*
DFSURGS0 (Database Scan utility)
overview 30
DFSURGU0 (HD Reorganization Unload utility) 301
DFSURPR0 (Database Prereorganization utility)
overview 262
DFSURRL0 (HISAM Reorganization Reload utility) 325
DFSURUL0 (HISAM Reorganization Unload utility)
overview 336
HALDB Index/ILDS Rebuild utility (DFSPREC0) 244
HALDB Migration Aid utility (DFSMAID0) 271
HALDB Partition Data Set Initialization utility (DFSUPNT0) 79
HALDB Partition Definition utility (%DFSHALDB) 87
HD Reorganization Reload utility (DFSURGL0) 283
HD Reorganization Unload utility (DFSURGU0) 301
HISAM Reorganization Reload utility (DFSURRL0) 325
HISAM Reorganization Unload utility (DFSURUL0)
overview 336
MSDB Dump Recovery utility (DBFDBDR0) 251
MSDB Maintenance utility (DBFDBMA0) 121
MSDB-to-DEDB Conversion utility (DBFUCDB0) 346
Online Database Image Copy utility (DFSUICP0) 174
Partial Database Reorganization utility (DFSPRCT1 and DFSPRCT2) 359
Program-Isolation-Trace Report utility (DFSPIRP0) 428
SB Test utility (DFSSBHD0)
overview 436
Utility Control Facility (DFSUCF00)
overview 378

V

VALUE= keyword
control statements
UCF FUNCTION=ZB 405
UCF FUNCTION=ZM 408
VERIFY= keyword
control statements
UCF FUNCTION=ZB 405
UCF FUNCTION=ZM 408
VSAM
load mode
HALDB Index/ILDS Rebuild utility 246
HD Reorganization Reload utility (DFSURGL0) 278
update mode
HALDB Index/ILDS Rebuild utility 246

VSAM *(continued)*
update mode *(continued)*
HD Reorganization Reload utility
(DFSURGL0) 278
VSAM (virtual storage access method)
UCF FUNCTION=SR control
statement 399

W

WF1DDS= keyword
UCF FUNCTION=DR control
statement 382
write-to-operator-with-reply (WTOR)
function
Utility Control Facility
(DFSUCF00) 417

Product Number: 5635-A02

Printed in USA

SC19-2435-05

Spine information:

IMS **Version 11** **Database Utilities** IBM