IMS
15.5.0

System Utilities
(2024-08-30 edition)

Note

Before you use this information and the product it supports, read the information in “Notices” on page 639.

2024-08-30 edition.

© Copyright International Business Machines Corporation 1974, 2024.
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
How new and changed information is identified..... vii
How to read syntax diagrams.....vii
Accessibility features for IMS 15.5.....ix
How to send your comments..... ix

Part 1. Generation utilities.....1

Chapter 1. Application Control Blocks Maintenance utility (DFSUACB0).....3
Examples of the ACB Maintenance utility.....11
Managing DOPT PSBs.....12
Managing DOPT PSBs in IMS systems that use ACB libraries.....13
Managing DOPT PSBs in IMS-managed ACB environments.....13

Chapter 2. Database Description (DBD) Generation utility..... 15
DBD generation for database types.....19
DBD generation input record structure (except for DEDB DBDs).....24
DEDB DBD generation input record structure.....26
DBD generation coding conventions..... 27
DBDGEN statements..... 27
DBD statements.....28
DATASET statements..... 48
AREA statement.....61
SEGM statements..... 63
LCHILD statements.....93
FIELD statements.....101
XDFLD statements.....123
DFSMARSH statements.....127
DFSMAP statements.....133
DFSCASE statements..... 136
DBDGEN, FINISH, and END statements.....140
Examples of the DBDGEN utility.....140
Examples without secondary indexes or logical relationships..... 140
Examples with logical relationships.....148
Examples with secondary indexes.....152
Running the DBDGEN procedure.....157

Chapter 3. MFS Device Characteristics Table utility (DFSUTB00).....161
Running the DFSUTB00 utility..... 164

Chapter 4. MFS Language utility (DFSUPAA0).....167
Utility control statements and syntax rules..... 172
Summary of control statements.....175
Message definition statements..... 177
Format definition statements.....188
Partition set definition statements.....240
Table definition statements.....243
Compilation statements..... 245
Running the utility in standard mode..... 251

Running the utility in batch mode....254
Running the utility in test mode....257
MFS library backup procedure....259
MFS restore procedure....260

Chapter 5. Program Specification Block (PSB) Generation utility....263
Utility control statements....266
Alternate PCB statement....266
Full-function or Fast Path database PCB statement....269
GSAM PCB statement....281
SENSEG statement....283
SENFLD statement....286
PSBGEN statement....287
END statement....291
Examples of the PSBGEN utility....291
Running the PSBGEN procedure....308

Part 2. IMS catalog utilities....311

Chapter 6. ACB Generation and Catalog Populate utility (DFS3UACB)....313

Chapter 7. IMS Catalog Alias Names utility (DFS3ALI0)....329

Chapter 8. IMS Catalog Copy utilities (DFS3CCE0, DFS3CCI0)....331
IMS Catalog Export utility (DFS3CCE0)....331
IMS Catalog Import utility (DFS3CCI0)....339

Chapter 9. IMS Catalog Directory Recovery utility (DFS3RU00)....353

Chapter 10. IMS Catalog Library Builder utility (DFS3LU00)....357

Chapter 11. IMS Catalog Maintenance utility (DFS3CM00)....365

Chapter 12. IMS Catalog Partition Definition Data Set utility (DFS3UCD0)....369

Chapter 13. IMS Catalog Populate utility (DFS3PU00)....373

Chapter 14. IMS Catalog Record Purge utility (DFS3PU10)....389

Part 3. Analysis utilities and reports....399

Chapter 15. Fast Path Log Analysis utility (DBFULTA0)....401
Fast Path report types....409

Chapter 16. File Select and Formatting Print utility (DFSERA10)....423
Examples of the DFSERA10 utility....432
DFSERA10 utility modules....438
Record Format and Print module (DFSERA30)....438
Program Isolation Trace Record Format and Print module (DFSERA40)....445
DL/I Call Image Capture module (DFSERA50)....448
IMS Trace Table Record Format and Print module (DFSERA60)....448
Enhanced Select module (DFSERA70)....449
OM Audit Trail Format and Print module (CSLULALE)....451

Chapter 17. IMS Monitor Report Print utility (DFSUTR20)....455
Examples of the DFSUTR20 utility....457

Chapter 18. Log Transaction Analysis utility (DFSILTA0)..... 459

Chapter 19. Offline Dump Formatter utility (DFSOFMD0).....465
Running the DFSOFMD0 utility..... 467

Chapter 20. Statistical Analysis utility (DFSISTS0)..... 471
Examples of the DFSISTS0 utility.....479

Part 4. Log utilities.....491

Chapter 21. Log Archive utility (DFSUARC0).....493
Examples of the DFSUARC0 utility..... 502

Chapter 22. Log Merge utility (DFSLTMG0)..... 503

Chapter 23. Log Recovery utility (DFSULTR0).....507
Examples of the DFSULTR0 utility..... 519

Part 5. Service utilities.....525

Chapter 24. Batch SPOC utility (CSLUSPOC)..... 527
Examples of the Batch SPOC utility.....530

Chapter 25. Database Recovery Control utility (DSPURX00)..... 531
Examples of the DSPURX00 utility..... 534
Invoking the utility using entry point DSPURXRT.....534

Chapter 26. Dynamic SVC utility (DFSUSVC0)..... 537
Examples of the DFSUSVC0 utility.....539

Chapter 27. Global Online Change utility (DFSUOLC0)..... 541
Examples of the DFSUOLC0 utility.....545

Chapter 28. MFS Service utility (DFSUTSA0).....547

Chapter 29. Multiple Systems Verification utility (DFSUMSV0)..... 561

Chapter 30. Online Change Copy utility (DFSUOCU0)..... 573
OLCUTL procedure..... 578
Initializing the IMS.MODSTAT data set.....579

Chapter 31. Spool SYSOUT Print utility (DFSUPRT0)..... 583
Examples of the DFSUPRT0 utility.....586

Chapter 32. Time-Controlled Operations Verification utility (DFSTVER0).....587
Examples of the DFSTVER0 utility..... 589

Part 6. Dynamic resource definition utilities..... 593

Chapter 33. Repository to RDDS utility (CSLURP20)..... 595
Examples of the CSLURP20 utility..... 597

Chapter 34. RDDS to Repository utility (CSLURP10)..... 599
Examples of the CSLURP10 utility..... 602

Chapter 35. Copy RDDS utility (DFSURCP0)..... 605

Examples of the DFSURCP0 utility..... 607

Chapter 36. Create RDDS from Log Records utility (DFSURCL0)..... 609
Examples of the DFSURCL0 utility.....614

Chapter 37. Create RDDS from MODBLKS utility (DFSURCM0)..... 619
Examples of the DFSURCM0 utility.....623

Chapter 38. DRD IMS SYSGEN stage 1 pre-parser utility (DFSURST0)..... 627
Examples of the DFSURST0 utility.....631

Chapter 39. RDDS Extraction utility (DFSURDD0)..... 633
Examples for the DFSURDD0 utility.....636

Notices.....639
Programming interface information.....640
Trademarks.....640
Terms and conditions for product documentation..... 641
IBM Online Privacy Statement..... 641

Bibliography.....643

Index..... 645

About this information

These topics provide reference information for the utilities that you can use with the IMS system to generate IMS resources, work with the IMS catalog, analyze IMS activity, manage IMS logging, run the IMS Database Recovery Control (DBRC) facility, maintain IMS networking services, and use dynamic resource definition (DRD).

This information is available in IBM® Documentation.

Prerequisite knowledge

Before using this information, you should understand z/OS®, and be familiar with IMS concepts, facilities, and access methods. The prerequisite publications are:

- *IMS Version 15.5 Communications and Connections*
- *IMS Version 15.5 Database Administration*
- *IMS Version 15.5 System Administration*

To learn about z/OS, see z/OS Basic Skills. For more resources, see IBM Z Education and Training.

To learn about IMS, see the IBM Press publication *An Introduction to IMS*, the resources listed for IBM Information Management System, and the variety of options available in IBM Training.

How new and changed information is identified

For most IMS library PDF publications, information that is added or changed after the PDF publication is first published is denoted by a character (revision marker) in the left margin. The *Program Directory* and *Licensed Program Specifications* do not include revision markers.

Revision markers follow these general conventions:

- 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.

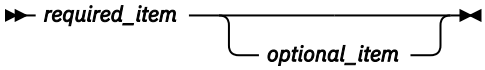
How to read syntax diagrams

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

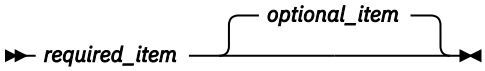
- Read the syntax diagrams from left to right, from top to bottom, following the path of the line. The following conventions are used:
  - The >>--- symbol indicates the beginning of a syntax diagram.
  - The ---> symbol indicates that the syntax diagram is continued on the next line.
  - The >--- symbol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

© Copyright IBM Corp. 1974, 2024

- Optional items appear below the main path.

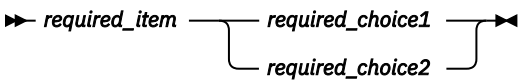


If an optional item appears above the main path, that item has no effect on the execution of the syntax element and is used only for readability.

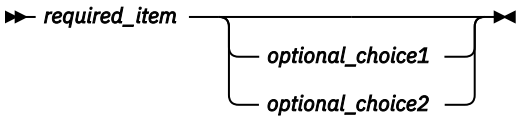


- If you can choose from two or more items, they appear vertically, in a stack.

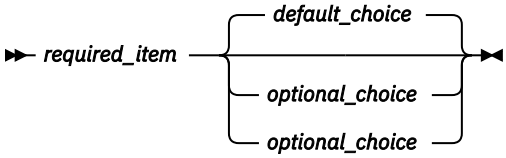
  If you *must* choose one of the items, one item of the stack appears on the main path.



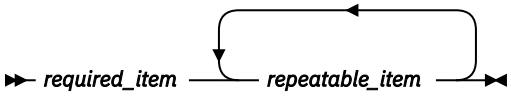
If choosing one of the items is optional, the entire stack appears below the main path.



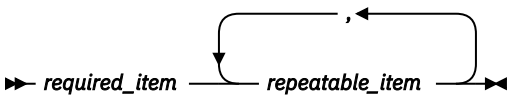
If one of the items is the default, it appears above the main path, and the remaining choices are shown below.



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



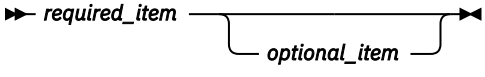
If the repeat arrow contains a comma, you must separate repeated items with a comma.



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 15.5

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 15.5. 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 15.5 ISPF panel functions by using a keyboard or keyboard shortcut keys.

For information about navigating the IMS 15.5 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 15.5 is available in IBM Documentation.

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

About this task

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:

Procedure

- Submit a comment by using the DISQUS commenting feature at the bottom of any IBM Documentation topic.
- Send an email to imspubs@us.ibm.com. Be sure to include the book title.
- Click the **Contact Us** tab at the bottom of any IBM Documentation topic.

What to do next

To help us respond quickly and accurately, please include as much information as you can about the content you are commenting on, where we can find it, and what your suggestions for improvement might be.

Part 1. Generation utilities

Use the generation utilities to generate and configure an IMS system.

Each topic introduces how the utility works, defines requirements and restrictions for its use, and provides examples.

© Copyright IBM Corp. 1974, 2024

Chapter 1. Application Control Blocks Maintenance utility (DFSUACB0)

In IMS systems that use ACB libraries to manage runtime application control blocks, use the Application Control Blocks (ACB) Maintenance utility to save instruction execution and direct-access wait time and improve performance in application scheduling.

This utility might not be used in IMS systems that manage runtime application control blocks by using the catalog. It provides a facility for pre-building the required application control blocks offline; so that when the application is scheduled, its application control blocks can be read directly, and control can be passed promptly to the application program.

When an application program is scheduled for execution, IMS must first have available database descriptor and PSB control blocks previously created. These control blocks can be created by the DBDGEN and PSBGEN procedures.

These control blocks must then be merged and expanded into an IMS internal format called *application control blocks* (ACBs). The merge and expansion process is called *block building*.

Application control blocks required for the DB/DC environment must be prebuilt, except for application programs that use a GPSB. It is optional for the batch environment. Using IMS.ACBLIB in a batch environment requires less virtual storage than building the ACBs dynamically from PSBLIB and DBDLIB.

The ACB Maintenance utility maintains the prebuilt blocks (ACB) library (IMS.ACBLIB). The ACB library is a consolidated library of program (PSB) and database (DBD) descriptions. Through control statements, you can direct the maintenance utility to build all control blocks for all PSBs, for a specific PSB, or for all PSBs that reference a specific DBD.

The ACB Maintenance utility does not populate the IMS catalog. To populate the IMS catalog after the ACB Maintenance utility builds the ACBs, use the IMS Catalog Populate utility (DFS3PU00).

As an alternative to running both the ACB Maintenance utility and the IMS Catalog Populate utility, you can use the ACB Generation and Catalog Populate utility (DFS3UACB), which builds the ACBs and populates the IMS catalog in a single job step.

Subsections:

- “Restrictions” on page 3
- “Prerequisites” on page 4
- “Requirements” on page 4
- “Recommendations” on page 5
- “Input and output” on page 5
- “JCL specifications” on page 6
- “Utility control statements” on page 7
- “Return codes” on page 11

Restrictions

You do not need to run ACB generation if your application program requires only an I/O PCB and one modifiable alternate PCB. Such applications, typically used in a DCCTL environment, can use GPSBs to define the resources necessary for execution.

You cannot predefine GSAM PSBs and DBDs using ACB generation because the control blocks for GSAM are different from the standard IMS data set control blocks. PSBs that reference GSAM, as well as non-GSAM databases, can be predefined using ACB generation to build the control block for the non-GSAM databases.

© Copyright IBM Corp. 1974, 2024

The ACB Maintenance utility uses some IMS system resources but not the total system. IMS.PSBLIB and IMS.DBDLIB are shared data sets. IMS.ACBLIB must be used exclusively. The utility can only be executed using an ACB library which is not concurrently allocated to an active IMS system.

IMS.ACBLIB is modified and cannot be used for any other purpose during execution of this program. IMS.ACBLIB is a partitioned data set and carries required linkage information in the directory. You can use the operating system (IEHMOVE) and data set (IEBCOPY) utilities for maintenance purposes.

Do not add FP DBDs to the active ACBLIB between an abnormal termination and /ERE. FP DBDs added to the active ACBLIB after abnormal termination of IMS are inaccessible after /ERE.

A Fast Path secondary index database supports only symbolic pointers. The ACB Maintenance utility issues message DFS2292E when PTR=SYMB is not specified on a LCHILD statement for a HISAM or SHISAM secondary index database. The primary DEDB database and its secondary index databases are deleted from the ACBLIB.

A user partition group for a Fast Path secondary index must contain all HISAM secondary index databases or all SHISAM secondary index databases in the same user partition group. The LCHILD statement contains both HISAM and SHISAM secondary index databases in the same user partition group identified in the DBD dbdname in the message. The primary DEDB database and its secondary index databases are deleted from the ACBLIB.

When a SENSEG statement for a segment that is other than a direct parent segment of the target segment along the physical path from the root segment or a child segment of the target segment in the PCB with the PROCSEQD operand is specified, the ACB Maintenance utility detects the invalid SENSEG statement specification. The ACB Maintenance utility issues a message DFS2295E. The PSB identified in message DFS2295E is deleted in the ACBLIB.

User partitioning is requested for Fast Path HISAM secondary index databases or Fast Path SHISAM secondary index databases. However, the user partition database specified in the PROCSEQD= parameter on the PCB statement is not the first user partition in the user partition group as defined in the NAME= parameter on the LCHILD statement in the primary DEDB database DBD. The ACB Maintenance utility issues message DFS2366E. The primary DEDB database and its secondary index databases are deleted in the ACBLIB.

A PSB has the PSELOPT= parameter specified on a PCB statement for a primary DEDB database and there is no user partitioning requested. The primary DEDB database has only one secondary index database specified in the NAME= parameter on a LCHILD statement in the primary DEDB DBD. The ACB Maintenance utility issues message DFS2367E. The PSB identified in the message is deleted in the ACBLIB.

Prerequisites

The ACB Maintenance utility does not change the PSB in IMS.PSBLIB or the DBD in IMS.DBDLIB. If changes are made in either PSBs or DBDs that require changes in the associated PSB or DBD, you must make these changes before running the utility. You can make additions, changes, and deletions to IMS.ACBLIB without stopping IMS, by using the Online Change utility and commands.

Changes in PSBs might also require modifications to the affected application programs. For example, if a DBD has a segment name changed, all PSBs which are sensitive to that segment must have their SENSEG statements changed.

Application programs which use this database might also need to be modified.

Requirements

IMS conforms to z/OS rules for data set authorization. If an IMS job step is authorized, all libraries used in that job step must be authorized. To run an IMS batch region as unauthorized, a non-authorized library must be concatenated to IMS.SDFSRESL.

Recommendations

If the IMS catalog is enabled in your IMS system, specify an output data set with the ACBCATWK DD statement so that the ACB Maintenance utility records a list of the ACB members it generates during the current execution. Providing this record of generated ACB members as input to the DFS3PU00 utility significantly reduces the time required to populate the IMS catalog.

Input and output

The following figure shows the functional relationship of the I/O data sets and their naming requirements. The ACB Maintenance utility receives input from IMS.DBDLIB data set, IMS.PSBLIB data set, SYSIN control statements, COMPCTL IEBCOPY control statements, and SYSPRINT messages. The ACB Maintenance utility outputs to the SYSUT3 and SYSUT4 IEBCOPY utility data sets, and the IMS.ACBLIB data set.

In IMS systems that have enabled the IMS catalog, the ACB Maintenance utility can optionally output a list of the generated ACB members to a data set referenced by the ACBCATWK DD statement. The DFS3PU00 utility reads the list of generated ACB members as input to significantly reduce the time required to populate the IMS catalog.

ACB generation procedure

The procedure is built during SMP/E processing and placed in the ADFSPROC and SDFSPROC data sets.

The following example shows the procedure for ACBLIB maintenance.

```
//       PROC SOUT=A,COMP=,RGN=4M,SYS2=,
//               NODE1=IMS,
//               NODE2=IMS
//G      EXEC PGM=DFSRRC00,PARM='UPB,&COMP',
//       REGION=&RGN
//SYSPRINT DD SYSOUT=&SOUT
//STEPLIB  DD DSN=&NODE2..&SYS2.SDFSRESL,DISP=SHR
//DFSRESLB DD DSN=&NODE2..&SYS2.SDFSRESL,DISP=SHR
//IMS      DD DSN=&NODE1..&SYS2.PSBLIB,DISP=SHR
//         DD DSN=&NODE1..&SYS2.DBDLIB,DISP=SHR
//IMSACB   DD DSN=&NODE1..&SYS2.ACBLIB,DISP=OLD
//SYSUT3   DD UNIT=SYSDA,SPACE=(80,(100,100))
//SYSUT4   DD UNIT=SYSDA,SPACE=(256,(100,100)),
//            DCB=KEYLEN=8
//COMPCTL  DD DISP=SHR,
//         DSN=&NODE2..&SYS2.PROCLIB(DFSACBCP)
```

In the figure, the high-level qualifier of the IMS data sets is IMS. This high-level qualifier is the default provided by IMS generation. However, if the default value was not used in IMS generation at your installation, the high-level qualifier for the IMS data set names might not be IMS.

ACB generation JCL statements

The following is a sample of the JCL statements that can be used to invoke the ACB generation procedure.

```
//ACBGEN   JOB
//         EXEC ACBGEN
//SYSIN    DD   *
 BUILD PSB=(MYPSB)
```

The ACB generation procedure uses the following symbolic variables:

SOUT=
Specifies the SYSOUT class. The default is A.

COMP=
PRECOMP,POSTCOMP, in any combination, cause the required in-place compression. The default is none.

RGN=
Specifies the region size for execution of the ACB utility. This region size depends on the size of the blocks to be generated and typically varies from 100 to 150 KB. The default is 4 MB.

SYS2=
Specifies an optional second-level dsname qualifier. When specified, the parameter must include a trailing period and be enclosed in quotes, for example:

```
SYS2='IMSA.'
```

JCL specifications

EXEC statement

The first part of the EXEC statement must be in the form:

```
PGM=DFSRRC00
```

A parameter field must be in the form:

```
PARM='UPB,PRECOMP,POSTCOMP'
```

where PRECOMP requests the IMS.ACBLIB data set be compressed before blocks are built, and POSTCOMP requests compression after the blocks are built. 'UPB' indicates that the block maintenance utility is to receive control. This parameter is required. PRECOMP and POSTCOMP are optional and can be used in any combination. Do not specify the POSTCOMP option if the ACBCATWK DD statement is used.

DD statements

ACBCATWK
Defines an optional work data set that contains a list of the ACB members that are written to the ACB library during ACB generation.

The ACBCATWK data set is an output data set for the ACB Maintenance utility and an input data set for the DFS3PU00 utility.

You must specify a single ACBCATWK data set. Multiple data sets are not supported by the utility.

Specify the ACBCATWK data set to improve the performance of the DFS3PU00 utility. The DFS3PU00 utility uses the list of names to determine which records in the IMS catalog need to be inserted or updated. If you do not specify the ACBCATWK data set, the DFS3PU00 utility processes all members in the ACB libraries that are referenced in the IMSACBxx DD statements.

COMPCTL DD
Defines the control input data set to be used by IEBCOPY if PRECOMP or POSTCOMP is specified.

If both PRECOMP and POSTCOMP are requested on the EXEC statement parameters, this data set must be capable of being closed with a reread option.

This data set must contain the following control statement of the form:

```
COPY  INDD=IMSACB,OUTDD=IMSACB
```

DFSRESLB DD
Points to an authorized library that contains the IMS SVC modules. For IMS batch, SDFSRESL and any data set that is concatenated to it on the DFSRESLB DD statement must be authorized through the Authorized Program Facility (APF). This DD statement provides an authorized library for the IMS SVC

modules, which must be in an authorized library. The JOBLIB or STEPLIB statement does not need to be authorized for IMS batch.

IMS DD
Defines the IMS.PSBLIB and IMS.DBDLIB data sets.

IMSACB DD
Defines a single ACB library data set.

Restriction: This data set is modified and cannot be shared with other jobs.

STEPLIB DD
Points to IMS.SDFSRESL, which contains the IMS nucleus and required IMS modules. If STEPLIB is unauthorized by having unauthorized libraries that are concatenated to IMS.SDFSRESL, you must include a DFSRESLB DD statement.

SYSIN DD
Defines the input control statement data sets. They can be on a tape volume, direct-access device, card reader, or be routed through the input stream. The input can be blocked as multiples of 80. During execution, this utility can process as many control statements as required.

SYSPRINT DD
Defines the output message data set.

When the SYSPRINT DD statement refers to a DASD or tape data set, you can control the block size of this data set with the BLKSIZE subparameter of the **DCB** parameter. If specified, the BLKSIZE value must be an exact multiple of 121 or a system ABEND013-20 can result. Omitting BLKSIZE from a DASD data set causes a system-determined block size to be used. Regardless of what value is specified for the LRECL parameter, the utility always uses a record length of 121.

SYSUT3 DD
Defines a work data set that is required if either PRECOMP or POSTCOMP is specified on the EXEC statement.

SYSUT4 DD
Same function as SYSUT3.

DFSACBCP control statement

The following control statement is created by using the DFSPROCB JCL that is built by SMP/E processing.

```
COPY  INDD=IMSACB,OUTDD=IMSACB
```

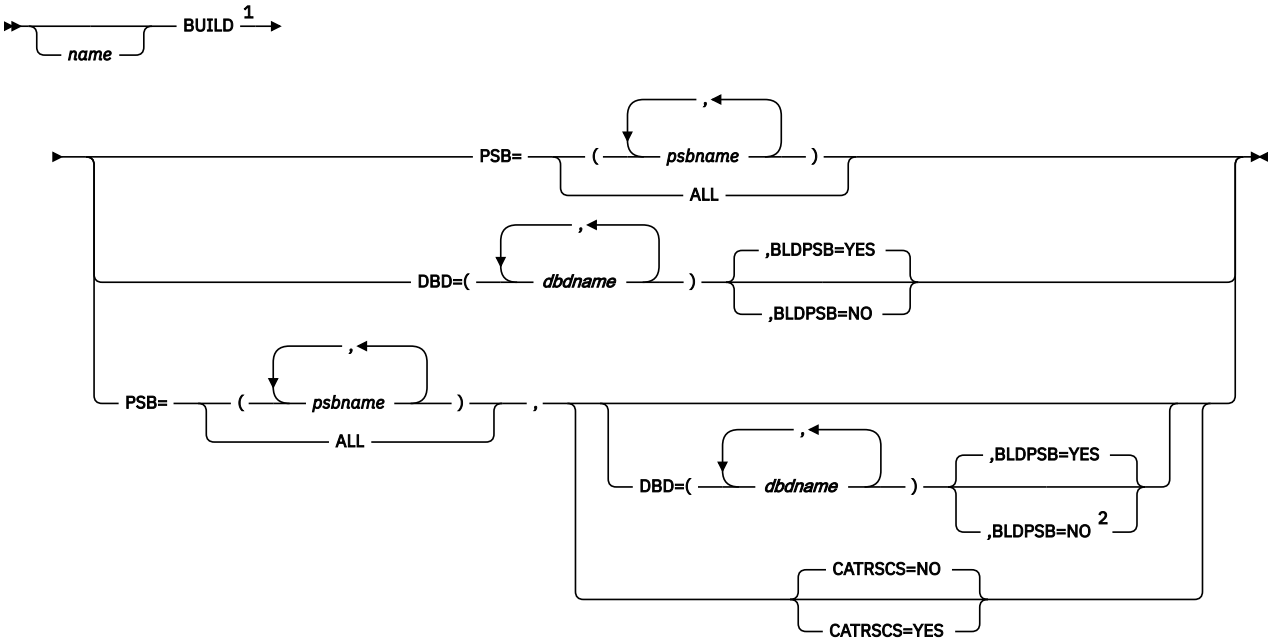
The ACB generation procedure uses DFSACBCP to compress ACBLIB.

Utility control statements

You specify control statements in the utility JCL to build or delete ACB members. The control statements must conform to the following guidelines:

- A statement is coded as a card image and is contained in columns 1 - 71.
- The control statement can optionally contain a name, starting in column 1.
- To continue a statement, enter a non-blank character in column 72 and begin the statement on the next line starting in column 16.
- The operation field must be preceded and followed by one or more blanks.
- The parameter is composed of one or more PSB or DBD names and must also be preceded and followed by one or more blanks.
- Commas, parentheses, and blanks can be used only as delimiting characters.
- Comments can be written following the last parameter of a control statement, separated from the parameter by one or more blanks.

ACB Maintenance utility syntax: BUILD format



Notes:

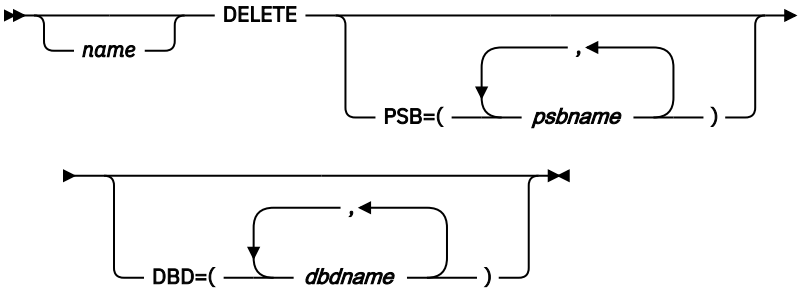
[1] There is no first in, first out (FIFO) process for the ACB Maintenance utility SYSIN input control statements. If both the BUILD PSB= and BUILD DBD= parameters are specified in the same application control block (ACB) generation job SYSIN control statement, DBD= operands are passed to the block builder utility program first. DFS0586I is issued if the DBD is not already in the ACBLIB data sets, regardless of where DBD= operands are entered in the SYSIN control statements.

[2] If you specify the parameters PSB=ALL and BLDPSB=NO in the same statement, IMS builds all of the PSBs (BLDPSB=NO is ignored). Similarly, if you specify the BLDPSB=NO parameter for one DBD and the BLDPSB=YES parameter on another DBD in the same ACBGEN job, IMS builds all the PSBs that refer to the changed DBDs and ignores the BLDPSB=NO specification. Also, if you specify BLDPSB=NO, no PSBs will be built for all SYSIN ACBGEN control cards.

In the following example, all of the PSBs that are associated with the CUSTOMER and ORDER DBDs are rebuilt, even though BLDPSB=NO is specified for the CUSTOMER DBD:

```
BUILD DBD=(CUSTOMER),BLDPSB=NO
BUILD DBD=(ORDER),BLDPSB=YES
```

ACB Maintenance utility syntax: DELETE Format



ACB Maintenance utility parameters

BUILD
Specifies that blocks are built for the named PSBs, which refer to the named DBDs.

DELETE
Specifies that blocks are deleted from the ACBLIB data set. The named PSBs and all PSBs that refer to the named DBDs are deleted.

Deleting a block from the ACBLIB data set does not delete the corresponding record in the IMS catalog.

PSB=ALL
Specifies that blocks are built for all PSBs that currently reside in IMS.PSBLIB. You use this parameter to create an initial IMS.ACBLIB. When the PSB=ALL parameter is specified, all PSBs and DBDs (and any other modules) are deleted from the ACBLIB data set and their space is available for reuse. Then an ACB generation is executed for every PSB in the PSBLIB data set. Do not use this parameter with a DELETE statement.

Restriction: When you specify the BUILD PSB=ALL parameter on a SYSIN control statement, all PSBs must reside in a single PSBLIB data set. No concatenated PSBLIBs are recognized on the IMS DD statement.

Restriction: Specifying PSB=ALL with CATRSCS=NO (or defaulting to CATRSCS=NO) does not build or update the IMS catalog PSBs and DBDs. If some PSBs or DBDs are not built or updated, message DFS5008W is issued and those PSBs or DBDs are ignored.

PSB=(*psbname*)
Specifies that blocks are built or deleted for all PSBs that are named on this control statement. As many of this type of control statement as required can be submitted. This parameter adds a new PSB to IMS.ACBLIB or delete a PSB no longer in use. You can omit the parentheses if you supply a single parameter.

DBD=(*dbdname*)
Specifies that blocks are built or deleted for this DBD and for all PSBs that reference this DBD either directly or indirectly through logical relationships. The DBD to be built must already exist in the IMS.ACBLIB data set. The referencing PSBs must already exist in the IMS.ACBLIB data set. PSBs that are newly added to the IMS.PSBLIB data set must be referenced by PSB operands. Because deleting a PSB does not delete any DBDs referenced by the PSB, this parameter can be used to delete specific DBDs. However, deleting or building a DBD causes every PSB in the IMS.ACBLIB data set that references the named DBD to be rebuilt or deleted based on the request type. You can omit the parentheses if you supply a single parameter.

Example 1: PSB-a references DBD-a and DBD-b. A DBDGEN was done for DBD-a and DBD-b and the updated DBDs are in DBDLIB (but not ACBLIB yet). By specifying DBD-a in an ACB generation, DBD-a is rebuilt in ACBLIB and any referencing PSBs (in this case PSB-a) are also rebuilt. Even though PSB-a has been rebuilt, the ACBLIB is not usable because DBD-b was not specifically rebuilt in ACBLIB. For DBD-b to be rebuilt in ACBLIB, it must be explicitly specified in the ACB generation. Although the referencing PSB is completely updated, the updated DBDs must be explicitly specified in the ACB generation.

Every PSB processed by this program generates a member in the IMS.ACBLIB data set. DBDs referenced by PSBs generate a member the first time the specific DBD is processed or any time a DBD name appears on a control statement. All PSBs that reference the same DBD carry information in their directory entries to connect the PSB to the referenced DBDs.

Logical DBDs do not have members in IMS.ACBLIB and cannot be referenced on BUILD or DELETE control statements.

Example 2: The following examples illustrate uses of the BLDPSB parameter:

- The DBD named CUSTOMER was changed and all of the PSBs that refer to CUSTOMER need to be rebuilt:

```
BUILD DBD=CUSTOMER,BLDPSB=YES
```

- The DBDs named ORDER and INVENTORY are changed and all of the PSBs that refer to these DBDs need to be rebuilt:

```
BUILD DBD=(ORDER,INVENTORY),BLDPSB=YES
```

When a DBD is replaced in IMS.DBDLIB, it must also be included in a BUILD DBD control statement. This is the only valid way the DBD can be replaced in IMS.ACBLIB without doing a BUILD PSB=ALL.

If a BUILD PSB is performed that references a modified DBD on DBDLIB, the PSB replaced on ACBLIB will contain the updated version of the DBD. If this BUILD PSB occurs before a BUILD DBD for the changed DBD, ACBLIB will contain PSBs with different versions of the DBD. The PSBs specified in the BUILD PSB will contain the updated DBD, while those not built will reference the old DBD. When a DBD for a PSB on ACBLIB does not match the accessed database, the results will be unpredictable. (For example, U852 abend occurs because segment codes have been added or deleted in the changed DBD). Therefore, when DBDGEN is run for later use, do not build a PSB that refers to the changed DBD unless the database reflects the change.

When a physical DBD is changed and is referenced in a BUILD DBD statement, all physical DBDs that are logically related to the one that was changed (including primary indexes and secondary indexes) must also be referenced in a BUILD DBD statement. However, DBDs that are logically related to these DBDs do not need to be rebuilt.

The following figure illustrates the relationships between some physical databases, where A is the changed DBD. The following relationships exist:

- B and C are logically related to A.
- D is logically related to B.
- E is logically related to C.
- D and E are not referenced in the BUILD DBD statement because they are not logically related to A.

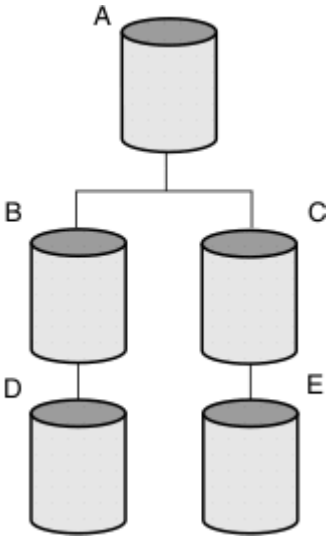


Figure 1. Example of logically related physical databases

BLDPSB=YES | NO

Specifies whether ACBGEN rebuilds all PSBs that reference a changed DBD in the BUILD DBD=(*dbdname*) statement.

YES

Indicates that ACBGEN rebuilds all PSBs that reference the changed DBD on the BUILD DBD=(*dbdname*) statement. The default is BLDPSB=YES.

NO

Indicates that ACBGEN does not rebuild PSBs that reference the changed DBD if the changed DBD does not change the physical structure of the database. For Fast Path DEDBs, the PSBs are rebuilt only when the number of segments, the number of fields within the segments of the database, or both are changed. For Fast Path MSDBs, the referencing PSBs are not rebuilt even if the database has physical structure changes.

CATRSCS=YES | NO
Specifies whether ACBGEN builds all PSBs and DBDs of the IMS catalog.

YES
Indicates that ACBGEN builds all IMS catalog PSBs and DBDs.

NO
Indicates that ACBGEN does not build any of the IMS catalog PSBs and DBDs. The default is CATRSCS=NO.

Return codes

The ACB generation procedure returns the following codes:

Code
Meaning

0
Successful completion of all operations

4
One or more warning messages issued

8
One or more blocks could not be built

16
Program terminated due to severe errors

Related concepts
Building the application control blocks (ACBGEN) (Database Administration)
Allocating ACBLIB data sets (System Definition)

Related reference
“ACB Generation and Catalog Populate utility (DFS3UACB)” on page 313
Use the ACB Generation and Catalog Populate utility (DFS3UACB) to generate ACB members in an IMS.ACBLIB data set, create the corresponding metadata records in the IMS catalog, and, if your IMS system manages ACBs, add the resulting ACBs as pending changes to the staging data set of the IMS catalog, all in a single job step.

“IMS Catalog Populate utility (DFS3PU00)” on page 373
Use the IMS Catalog Populate utility (DFS3PU00) to load, insert, or update DBD and PSB instances into the database data sets of the IMS catalog from ACB library data sets. If the IMS management of ACBs is enabled in the IMS system, the utility also adds the active application control blocks (ACBs) to the IMS directory data set that is managed by IMS.

Examples of the ACB Maintenance utility

These examples show how to use the ACB Maintenance utility to create or delete blocks for PSBs.

Example of creating blocks for all PSBs

In this example, all blocks currently existing in IMS.ACBLIB are deleted and their space is reused to create new blocks for all PSBs that currently reside in IMS.PSBLIB. This option will normally be used for initial creation of the IMS.ACBLIB data set. If space is not yet allocated for ACBLIB, there should be a space parameter and a DISP=NEW on the IMSACB DD statement.

```
//BLDBLKS  JOB
//*
//STEP    EXEC ACBGEN,SOUT=A
//SYSIN     DD *
            BUILD PSB=ALL
/*
```

Example of creating blocks for specific PSBs

This example creates blocks for PSB1, PSB2, and PSB3. All other PSBs in IMS.ACBLIB remain unchanged. If any DBDs referenced by these PSBs do not exist in IMS.ACBLIB, they are added. In addition, DBD5 and DBD6 are deleted from ACBLIB. IMS.ACBLIB is compressed after the blocks are built, and deletions are performed.

```
//BLDBLKS  JOB
//*
//STEP    EXEC ACBGEN,SOUT=A,COMP=POSTCOMP
//SYSIN     DD *
            BUILD  PSB=(PSB1,PSB2,PSB3)
            DELETE DBD=(DBD5,DBD6)
/*
```

Example of deleting a PSB and rebuilding blocks

This example deletes PSB1 from the IMS.ACBLIB data set and causes all PSBs in the IMS.ACBLIB data set that reference DBD4 to have their blocks rebuilt. If PSB1 referenced DBD4, it will not be rebuilt, since PSB1 had just been deleted from IMS.ACBLIB. PSB1 is not deleted from IMS.PSBLIB. The IMS.ACBLIB is compressed before and after the blocks have been built.

```
//BLDBLKS  JOB
//*
//STEP    EXEC ACBGEN,SOUT=A,COMP='PRECOMP,POSTCOMP'
//SYSIN     DD *
     DELETE PSB=PSB1
     BUILD DBD=DBD4
  /*
```

Managing DOPT PSBs

How you manage dynamic option (DOPT) PSBs that are generated by the ACB Maintenance utility differs depending on whether your IMS systems use ACB libraries or the IMS catalog as the repository for active ACBs.

In IMS systems that use an ACB library, the blocks for the DOPT PSBs must be contained in an ACBLIB data set concatenated after the primary ACBLIB data set.

In IMS systems that use the IMS catalog, the blocks for the DOPT PSBs are stored in the IMS catalog along with all the other active ACB blocks. You do not need to concatenate any data sets.

The PSB definitions and the blocks that are generated for DOPT PSBs are no different than the blocks for non-DOPT PSBs. The PSBs are defined as dynamic by the DOPT parameter in either an APPLCTN stage-1 system definition macro or in the IMS type-2 command **CREATE PGM** or **UPDATE PGM**.

If the programs that are associated with the PSBs are scheduled for a BMP or JBP dependent region, you can also use the Program Creation user exit (PGMCREAT) to define the PSBs as dynamic by setting the PGMCR_PF1_DOPTY bit in the PGMCREAT parameter list.

Related concepts

Declaring online application programs (System Definition)

Related reference

“IMS Catalog Populate utility (DFS3PU00)” on page 373

Use the IMS Catalog Populate utility (DFS3PU00) to load, insert, or update DBD and PSB instances into the database data sets of the IMS catalog from ACB library data sets. If the IMS management of ACBs is enabled in the IMS system, the utility also adds the active application control blocks (ACBs) to the IMS directory data set that is managed by IMS.

APPLCTN macro (System Definition)

CREATE PGM command (Commands)

PGMCREAT user exit routine type (Exit Routines)

Managing DOPT PSBs in IMS systems that use ACB libraries

Using dynamic option (DOPT) PSBs in IMS systems that use ACB library (ACBLIB) data sets requires concatenation of a primary ACBLIB data set. The first or primary data set in the ACBLIB data set concatenation should contain the blocks for all non-dynamic (non-DOPT) PSBs. A subsequent DOPT ACBLIB data set should contain blocks for all dynamic option (DOPT) PSBs.

Note: You must ensure that the active and inactive DOPT ACBLIB data sets have different names to ensure that online change finds the changes made to the DOPT ACBLIB data sets.

The primary ACBLIB data set is the first DD statement of the concatenation. To BUILD a PSB or DBD into the concatenated data sets, supply only one DD statement to the ACB Maintenance utility.

At system initialization time, all non-dynamic PSBs and all DBDs must have been built into either the primary or DOPT ACBLIB data sets.

By transaction schedule time, the DOPT PSBs being scheduled must be built into the DOPT ACBLIB data sets. Never build DOPT PSBs into the primary ACBLIB data sets.

If all PSBs in the system are DOPT PSBs, the primary ACBLIB should be a dummy PDS data set. The DOPT ACBLIB should contain blocks for all DBDs and PSBs. Set the DIRCA size parameter in the BMP, MPP, or IFP JCL.

If some, but not all, PSBs in the system are DOPT PSBs, both ACBLIB data sets will contain blocks for DBDs and PSBs. When you BUILD a PSB into one ACBLIB data set, the blocks for the DBDs referenced by the PSB are also built into that data set. If the DBD was already built into another ACBLIB data set, you will have two sets of blocks for the DBD. When DL/I does a BLDL to use the blocks for the DBD, it uses the set of blocks in the primary ACBLIB.

During the termination process of a program using DOPT PSBs, the PSBs are deleted from the PSB pool.

Related reference

DBLDL= parameter for procedures (System Definition)

Managing DOPT PSBs in IMS-managed ACB environments

When the IMS management of ACBs is enabled, the blocks for dynamic option (DOPT) PSBs that are generated by the ACB Maintenance utility are stored in IMS in the same way as the blocks for other types of PSBs: in the IMS catalog. IMS does not use an ACB library or require you to put the DOPT PSBs in a separate, concatenated data set.

About this task

You add ACB blocks for new or modified DOPT PSBs that are generated by the ACB Maintenance utility to the IMS catalog by providing the ACBLIB data set that contains them as input to the IMS Catalog Populate utility (DFS3PU00).

If the input ACBLIB contains only DOPT PSBs, the IMS Catalog Populate utility can make the DOPT PSBs available for immediate use without requiring you to issue the IMPORT DEFN command or restart IMS.

If the input ACBLIB contains DBDs and non-DOPT PSBs with the DOPT PSBs, you need to activate the DOPT PSBs with the other resource types by issuing the IMPORT DEFN SOURCE(CATALOG) command or, if the IMS Catalog Populate utility is run with the UPDATE option, by restarting IMS.

Procedure

To add DOPT PSBs only:

1. Generate the DOPT PSBs into an ACBLIB data set
2. Remove any blocks for DBDs or non-DOPT PSBs from the ACBLIB data set
3. Prepare the JCL for the IMS Catalog Populate utility by specifying:
   a) The ACBLIB that contains the DOPT PSBs on the IMSACB01 DD statement.

If you concatenate ACBLIB data sets, do not include any ACBLIB data sets that contain DBDs or non-DOPT PSBs.

b) The **MANAGEDACBS** control statement with the **UPDATE** and **SHARE** options.

If the **SHARE** option is specified when the input ACBLIB contains DBDs or non-DOPT PSBs, these resources are not activated in the IMS systems. No IMS system can use the new resources until either the system is restarted or some other event causes the system to load the resources into memory.

Before you can use the **SHARE** option, your z/OS system must be configured to support extended sharing of PDSE data sets. Specify **PDSESHARING(EXTENDED)** in the IGDSMS*xx* member of the z/OS SYS1.PARMLIB data set.

4. Run the IMS Catalog Populate utility.

When the **UPDATE** and **SHARE** options are specified, the IMS Catalog Populate utility adds the ACBs for the DOPT PSBs to the IMS catalog directly without staging the resources first or requiring you to issue the **IMPORT DEFN SOURCE(CATALOG)** command. The DOPT PSBs are added to the IMS catalog data sets in a shared mode that allows the online IMS systems to continue to access the IMS catalog while the IMS Catalog Populate utility adds the DOPT PSBs.

Related tasks

z/OS: Specifying Extended PDSE Sharing in a Multiple-System Environment

Related reference

IMS Catalog Populate utility (DFS3PU00) (System Utilities)

Chapter 2. Database Description (DBD) Generation utility

In IMS systems that use DBD libraries, use the Database Description Generation (DBDGEN) utility to define a database so that it can be used by an application program.

This utility might not be used in IMS systems that manage runtime application control blocks by using the catalog.

A database description (DBD) is a DL/I control block containing all of the database information needed by an application program.

You create a database description (DBD) by coding special macro instructions. These macros become the input to the DBDGEN utility.

You can use only one physical DBD to describe each physical database; otherwise, a user abend, such as 0850, 0852, or 0853 occurs. At execution time, DL/I uses the DBD to create a set of internal control blocks.

The DBDGEN utility defines each DBD with the following database information:

- Segment types
- Physical and logical relationships between segment types
- Database organization and access method
- Physical characteristics of the database
- Define the name and data options of selected exit routines
- Metadata that describes the database and the data stored in the database

Subsections:

- “Restrictions” on page 15
- “Prerequisites” on page 15
- “Requirements” on page 15
- “Recommendations” on page 15
- “Input and output” on page 16

Restrictions

Currently, no restrictions are documented for the DBDGEN utility.

Prerequisites

Currently, no prerequisites are documented for the DBDGEN utility.

Requirements

There are strict rules for structuring DBDGEN input. A separate input set is required for each database.

Recommendations

Currently, no recommendations are documented for the DBD Generation utility.

© Copyright IBM Corp. 1974, 2024

Input and output

The DBDGEN program accepts several types of control statements.

- The DBD statement names the database being described and provides DL/I with information concerning database organization.
- The DATASET statement is used only in non-DEDB DBDGEN input record structures. The DATASET statement defines a data set group within a database. One or more DATASET statements follow the DBD statement.
- The AREA statement is used only in DEDB DBDGEN input record structures. The AREA statement defines an area within a database. One or more AREA statements follow the DBD statement.
- The SEGM statement defines the segments of the specified database. The SEGM statement is used with the following statements:
  - FIELD
  - XDFLD
  - LCHILD
  - DFSMARSH
  - DFSMAP
  - DFSCASE

  Each statement defines different aspects of a segment or the fields in a segment.
- The DBDGEN statement indicates the end of DBDGEN control statements.
- FINISH is an optional statement retained in the input stream for compatibility.
- The END statement indicates to the z/OS assembler that the end of the input statements has been reached.

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

Control statement listing

This is a listing of the input statement images to this job step.

Diagnostics

Errors discovered during the processing of each statement result in diagnostic messages. These messages are printed immediately following the image of the last statement that is read. The message can reference either the statement immediately preceding it or the preceding group of statements. It is also possible that more than one message could be printed for each statement.

In this case, these messages follow each other on the output listing. After all the statements have been read, a further check is made of the reasonableness of the entire deck. This might result in one or more additional diagnostic messages.

Any discovered error results in the diagnostic messages being printed, the statements being listed, and the other outputs being suppressed. However, all the statements are read and checked before the DBD generation execution is terminated. The bind step of DBD generation is not processed if a statement error has been found.

Assembler listing

An assembler language listing of the DBD macro expansion created by DBD generation execution is provided. You can eliminate a printout of this listing by including an assembler language PRINT NOGEN statement.

If the DBD generation is for a database that uses VSAM as the operating system access method, a page in the assembler listing will provide recommended values for some of the parameters necessary to define the data sets of the database to VSAM. CONTROLINTERVALSIZE and RECORDSIZE values other than those recommended might be desired for special reasons, such as performance improvement. RECORDSIZE needs to be changed appropriately for all ESDS definitions.

If the control interval size is not specified (see the SIZE parameter in the GSAM row in Table 4 on page 54), it defaults to the size recommended in this assembler listing. The following example shows the output produced for a HISAM database. The parameters provided are in the format required for Access Method Services statements. The first DEFINE provides parameters for the key sequenced data set (KSDS) and the second DEFINE provides parameters for the entry sequenced data set (ESDS).

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

Example of Access Method Services parameters from DBD generation

If you use the **/DBD** command to allow an offline dump of a VSAM database, you must use SHARE OPTIONS(3) in the VSAM DEFINE operation for the data sets of the database. The following figure shows an example of Access Method Services parameters from DBD generation.

```
++,* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
++,*
++,     RECOMMENDED VSAM DEFINE CLUSTER PARAMETERS
++,*
++,* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
++,* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
++,*                  *NOTE 1
++,     DEFINE CLUSTER (NAME(DDI3I1) -
++,*           INDEXED KEYS (6, 10) -
++,*           RECORDSIZE (680,680) -
++,*           DATA (CONTROLINTERVALSIZE (4096))
++,*  *NOTE 1: SHOULD SPECIFY DSNAME FOR DDI3I1
++,* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
++,* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
++,*                  *NOTE 2
++,*    DEFINE CLUSTER (NAME(DDI3O1) NONINDEXED -
++,*           RECORDSIZE (680,680) -
++,*           CONTROLINTERVALSIZE (4096))
++,*  *NOTE 2: SHOULD SPECIFY DSNAME FOR DDI301
++,* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

Segment flag codes

Segment flags are printed in DBD generation output to confirm what has been generated by that particular DBD generation. The flags, when interpreted, tell you which pointer options were generated; the segment insert, delete, and replace rules specified; whether physical child pointers have been reserved in this segment's prefix; and how many physical children are related to the segment. Segment flags appear in the output as an assembler language defined constant (DC) statement. The constant is defined as 8 hexadecimal digits followed by the comment, SEGMENT FLAGS. Each pair of digits in the constant is a hexadecimal byte. To interpret the constant, convert the first 6 digits to binary values, and the last 2 digits to decimal values as shown in the following figure.

```
         CONVERTED
BYTE     VALUE         DESCRIPTION

 0                     POINTER POSITIONS GENERATED:

         1.......      CTR (Counter)
         .1......      Physical twin forward
         .11.....      Physical twin forward and backward
         ...1....      Physical parent
         ....1...      Logical twin forward
         ....11..      Logical twin forward and backward
         ......1.      Logical parent
         .1.....1      Hierarchic forward
```

```
         .11....1      Hierarchic forward and backward

 1                     SEGMENT PROCESSING RULES:

         10......      Insert physical
         01......      Insert virtual
         11......      Insert logical
         ..10....      Insert nonsequential last
         ..01....      Insert nonsequential first
         ..11....      Insert nonsequential here at current position
         ....10..      Replace physical
         ....01..      Replace virtual
         ....11..      Replace logical
         ......10      Delete physical
         ......01      Delete virtual
         ......11      Delete logical
         ......00      Bivirtual delete

 2       ..XX.XXX      Reserved
       1.......      Segment is paired
         .1......      Segment is a direct dependent in a FP DEDB
         ....1...      Segment's parent has two physical child
                       pointers; hierarchic pointers were not specified

 3       0-254         Number of physical children of this segment
                       pointed to by physical child pointers
```

Segment prefix format description

Convert the values to binary and decimal representations:

```
   Byte 0   Byte 1    Byte 2    Byte 3
    FE       FD        08       0A
11111110  11111101  00001000   10
```

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

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

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

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

Output from DBD generation contains the statement:

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

Load module

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

DBD generation error conditions

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

- DBD generation issues diagnostic messages that:
  - Flag operands or parameters that are not shown for the type of database being defined
  - Indicate that operands or parameters that are required for the type of database being defined were omitted

- DBD generation completes, but DL/I ignores the control information that was generated by the specification of operands or parameters that are not shown for the type of database that was defined.
- DBD generation completes, but DL/I is unable to create and access the defined database because (a) conflicting control information was specified when attempting to interrelate databases, or (b) segment relationships describing the application program's view of the database were not properly defined in the DBD generation.
- DBD generation completes, and DL/I creates and accesses a database. However, the results provided to you are not those you desired. This condition can occur because the default actions taken by DL/I in response to finding missing or conflicting control information are actions that you had not considered during DBD generation.

Related concepts

Coding database descriptions as input for the DBDGEN utility (Database Administration)

Building the application control blocks (ACBGEN) (Database Administration)

Allocating ACBLIB data sets (System Definition)

DBD generation for database types

The DBDGEN utility generates DBDs for a database based on the type of database that is using the utility.

The following types of databases use the DBDGEN utility:

- HSAM (including SHSAM)
- GSAM
- HISAM (including SHISAM)
- HDAM
- PHDAM
- HIDAM
- PHIDAM
- MSDB
- DEDB
- Index
  - Primary HIDAM
  - Secondary
- PSINDEX
- Logical

HSAM/SHSAM DBD generation

During DBD generation for an HSAM database, you specify:

- One data set group.
- The ddname of an input data set that is used when an application retrieves data from the database.
- The ddname of an output data set that is used when loading the database.
- From 1 to 255 segment types for the database.
- From 0 to 255 fields within each segment type, with a maximum of 1000 fields within the database.

For an HSAM database, you cannot specify:

- The use of hierarchic or physical child or physical twin pointers between segments in the database
- The use of logical or index relationships between segments

Optionally, you can define a simple HSAM (SHSAM) database that can contain only one fixed-length segment type. In this case, no prefixes are built in occurrences of the segment type.

During DBD generation for an SHSAM database, you specify:

- One data set group.
- The ddname of an input data set that is used when an application retrieves data from the database.
- The ddname of an output data set that is used when loading the database.
- From 0 to 255 fields within the single segment type.

GSAM DBD generation

During DBD generation for a GSAM database, you specify:

- One data set group.
- The ddname of an input data set that is used when an application retrieves data from the database.
- The ddname of an output data set that is used when loading the database.
- Optionally, a SEGM statement that describes a virtual segment.
- Optionally, a FIELD statement that is defined with an external name only. These fields are not searchable by IMS and therefore cannot be specified in an SSA. They are typically used to store field metadata from a COBOL copybook that would otherwise not be defined in IMS.

You cannot specify:

- A SEGM statement that describes a physical segment
- A FIELD statement that describes an IMS-searchable field
- The use of logical or index relationships between segments

For variable length GSAM/BSAM database, IMS adds 2 bytes to the record length value in the GSAM records passed by the application. This is done in order to accommodate the ZZ field that makes up the BSAM Record Descriptor Word (RDW) when the record is written to the I/O device.

The following figure shows that the four GSAM records (IMS segments) fit exactly in one 32,760 byte block.

```
//IDASD    DD DUMMY
//ODASD    DD UNIT=SYSDA,VOL=SER=000000,DISP=(,KEEP),
//         SPACE=(TRK,(5,1)),DSN=GSAM.VARIABLE1,
//         DCB=(RECFM=VB,BLKSIZE=32760,LRECL=32756)
//SYSIN    DD *,DCB=BLKSIZE=80
S 1 1 1 1 1    DBDNAME
L        ISRT
L  V8187 DATA  1ST RECORD LOADED TO GSAM
L        ISRT
L  V8187 DATA  2ND RECORD LOADED TO GSAM
L        ISRT
L  V8187 DATA  3RD RECORD LOADED TO GSAM
L        ISRT
L  V8187 DATA  4TH RECORD LOADED TO GSAM
```

HISAM/SHISAM DBD generation

During DBD generation for a HISAM or SHISAM database, you specify:

- One data set group.
- The ddname of one VSAM key sequenced data set (KSDS) and one VSAM entry sequenced data set (ESDS). HISAM supports only one data set group; you cannot have a secondary data set group with HISAM databases.

- Optionally, you can define a simple HISAM (SHISAM) database that can contain only one fixed-length segment type. In this case, no prefixes are built in occurrences of the segment type. The logical record length specified for a SHISAM database must be equal to or greater than the segment length specified.
- At least one segment type and a maximum of 255 segment types for the database.
- From 0 to 255 fields for each segment type, and a maximum of 1000 for the database, one of which must be a unique sequence field in the root segment type for indexing root segment occurrences.
- A maximum of 32 secondary index relationships (optional) per segment type, and a maximum of 1000 for the database.
- Logical relationships (optional) using symbolic pointer options when a segment in a HISAM database points to another segment in a HISAM database, and direct or symbolic pointer options when a segment in a HISAM database points to a segment in an HDAM or HIDAM database.
- Segment Edit/Compression exit routine routines, which are optional, to enable user-supplied routines to manipulate each occurrence of a segment type to or from auxiliary storage.
- Data Capture exit routine, which is optional, to enable Db2 for z/OS users access to updated IMS data. This exit routine can be used in SHISAM also.

Restriction: You cannot specify the use of hierarchic or physical child or physical twin pointers between segments in a HISAM database.

HDAM/PHDAM DBD generation

During DBD generation for HDAM and PHDAM databases, you specify:

- The name of the user-supplied randomizing module used for placement of root segment occurrences
- One to 10 data set groups
- How free space is to be distributed in each data set group
- The ddname of an OSAM or ESDS data set for each data set group defined
- At least one segment type for each data set group, and a maximum of 255 segment types for the database
- Segment Edit/Compression exit routine routines, which are optional, to enable user-supplied routines to manipulate each occurrence of a segment type on their way to or from auxiliary storage
- The use of hierarchic or physical child or physical twin pointers between segments in the database
- Logical relationships (optional) between segments using direct address or symbolic pointer options
- From 0 to 255 fields for each segment type, and a maximum of 1000 for the database
- A maximum of 32 secondary index relationships (optional) per segment type and a maximum of 1000 for the database
- Data Capture exit routine, which is optional, to enable Db2 for z/OS users access to updated IMS data

DBDGEN for PHDAM

- The ddnames and data sets are not part of DBDGEN for PHDAM databases. The remaining database definition is purely for defining the hierarchical structure and relationships of the data.
- DBDGEN does not define each individual partition.

HIDAM and PHIDAM DBD generation

During DBD generation for HIDAM and PHIDAM databases, you specify:

- One to 10 data set groups
- How free space is to be distributed in each data set group
- The ddname of an OSAM or ESDS data set for each data set group defined (HDAM databases only)

- At least one segment type for each data set group, and a maximum of 255 segment types for the database
- Segment Edit/Compression exit routine routines, which are optional, to enable user-supplied routines to manipulate each occurrence of a segment type on their way to or from auxiliary storage
- A maximum of 32 secondary index relationships (optional) per segment type and a maximum of 1000 for the database
- The use of hierarchic or physical child or physical twin pointers between segments in the database
- Logical relationships (optional) between segments using direct address or symbolic pointer options
- From 0 to 255 fields for each segment type, and a maximum of 1000 for the database, one of which must be a unique sequence field in the root segment type for indexing root segment occurrences
- Data Capture exit routine, which is optional, to enable Db2 for z/OS users access to updated IMS data

DBDGEN for PHIDAM:

- The ddnames and data sets are not part of DBDGEN for PHIDAM databases. The remaining database definition is purely for defining the hierarchical structure and relationships of the data.
- DBDGEN does not define each individual partition.

MSDB DBD generation

During DBD generation for an MSDB, you must specify:

- One database name
- One data set group
- One segment type for the database
- From 0 to 255 fields within the database

You cannot specify:

- A logical or index relationship between segments
- Fields used with secondary indexes
- Fields defined as arrays or structures

If the DBD for an existing MSDB is changed, the header information (BHDR) might change, even though the database segments are unchanged. This might result in message DFS2593I because of the attempted load from the MSDBCPx data set. In this case, the headers in the MSDBCPn data sets are either invalid or the wrong length. If ABND=y is specified in the MSDB PROCLIB member, it also causes a U1012 abend. After modifying the DBD, load the MSDBs from an MSDBINIT data set by using the MSDBLOAD option for either a warm start or a cold start to eliminate these problems.

DEDB DBD generation

During DBD generation for a DEDB, you must specify:

- One database name
- From 1 to 2048 areas within a database
- From 1 to 127 segment types for the database
- From 0 to 255 fields for each segment type, with a maximum of 1000 fields within the database, one of which must be a unique sequence field for the root segment type
- The ddname or area name used to describe an area
- Data Capture exit routine, which is optional, to enable Db2 for z/OS users access to updated IMS data

You can optionally specify up to eight subset pointers for each child type of the parent.

You cannot specify a logical or index relationship between segment types.

Index, PSINDEX DBD, and FPINDEX DBD generation

Primary HIDAM index DBD generation creates an index database composed of one index segment type that indexes occurrences of the HIDAM root segment type. PHIDAM does not have a DBD for the primary index. An index segment contains:

- The sequence field key of the root segment occurrence it indexes
- In its prefix, a direct address pointer to the root segment occurrence

During DBD generation for a primary HIDAM index, you must specify:

- One database name.
- One data set group. You must specify the ddname of one KSDS.
- One segment type.
- The index relationship required between the primary HIDAM index database and the root segment type of a HIDAM database.
- One field within the segment type as a sequence field.

Restriction:

- You cannot specify any additional FIELD statements as you might for a secondary index.
- You cannot use DBDGEN to define individual partitions.
- Non-unique secondary index (PSINDEX) databases are not supported for HALDB.

Secondary index DBD generation creates a secondary index database made up of 1 to 16 index pointer segment types. These are used to index target segment types in HISAM, SHISAM, HDAM, PHDAM, HIDAM, or PHIDAM databases.

During DBD generation for a full-function secondary index, you must specify:

- One database name.
- One data set group. If all index pointer segment keys are unique, you must specify the ddname of one KSDS. If index pointer segment keys are non-unique you must specify the ddnames of one KSDS and one ESDS. A secondary index must use VSAM.
- One segment type.
- One field for each segment type.

For a Fast Path secondary index, the DBD statement specifies the name of the secondary index database in the NAME= parameter. The ACCESS= parameter must specify one of the following values:

ACCESS=(INDEX,VSAM),FPINDEX=YES
A HISAM secondary index database on the DBD statement for the new Fast Path secondary index database.

ACCESS=(INDEX,SHISAM),FPINDEX=YES
A SHISAM secondary index database on the DBD statement for the new Fast Path secondary index database.

To define a primary DEDB database with secondary indexing, add LCHILD and XDFLD statements for the indexing fields in the DBD of the primary DEDB database.

If a HISAM secondary index database or a SHISAM secondary index database has two or more user partition databases defined in the NAME= parameter on the LCHILD statement, specify a user partition selection exit in the PSELRTN= parameter on an XDFLD statement in the primary DEDB database. The sample user partition selection exit is DBFPSE00. The PSELOPT=MULT|SNGL parameter can be specified on an XDFLD statement or on a PCB statement with the PROCSEQD= parameter to control whether a

single user partition or multiple user partitions are used before a GB status code is returned to indicate when the end of data is reached. The PSELOPT=MULT|SNGL must be explicitly specified on the PCB statement with the PROCSEQD= parameter. There is no default of PSELOPT=MULT on the PCB statement because its value overrides the PSELOPT=MULT|SNGL on the XDFLD statement.

Logical DBD generation

A logical DBD generation creates a logical database made up of logical segment types. A logical segment type is a segment type defined in a logical database that represents a segment type or the concatenation of two segment types defined in a physical database or databases.

During DBD generation for a logical database, you must specify:

- One database name.
- One logical data set group.
- From 1 to 255 segment types. Each defines the name of a logical segment type, and the name of the segment type or types in physical databases that are to be processed when a call is issued to process the logical segment type.

The logical relationships used to create a logical database must be defined in a physical database or databases.

All fields required for segments in a logical database must have been defined in physical databases.

DBD generation input record structure (except for DEDB DBDs)

The DBDGEN program accepts control statements that must be added to the SYSIN input stream in a specific order.

The following figure shows the rules for structuring DBD generation input.

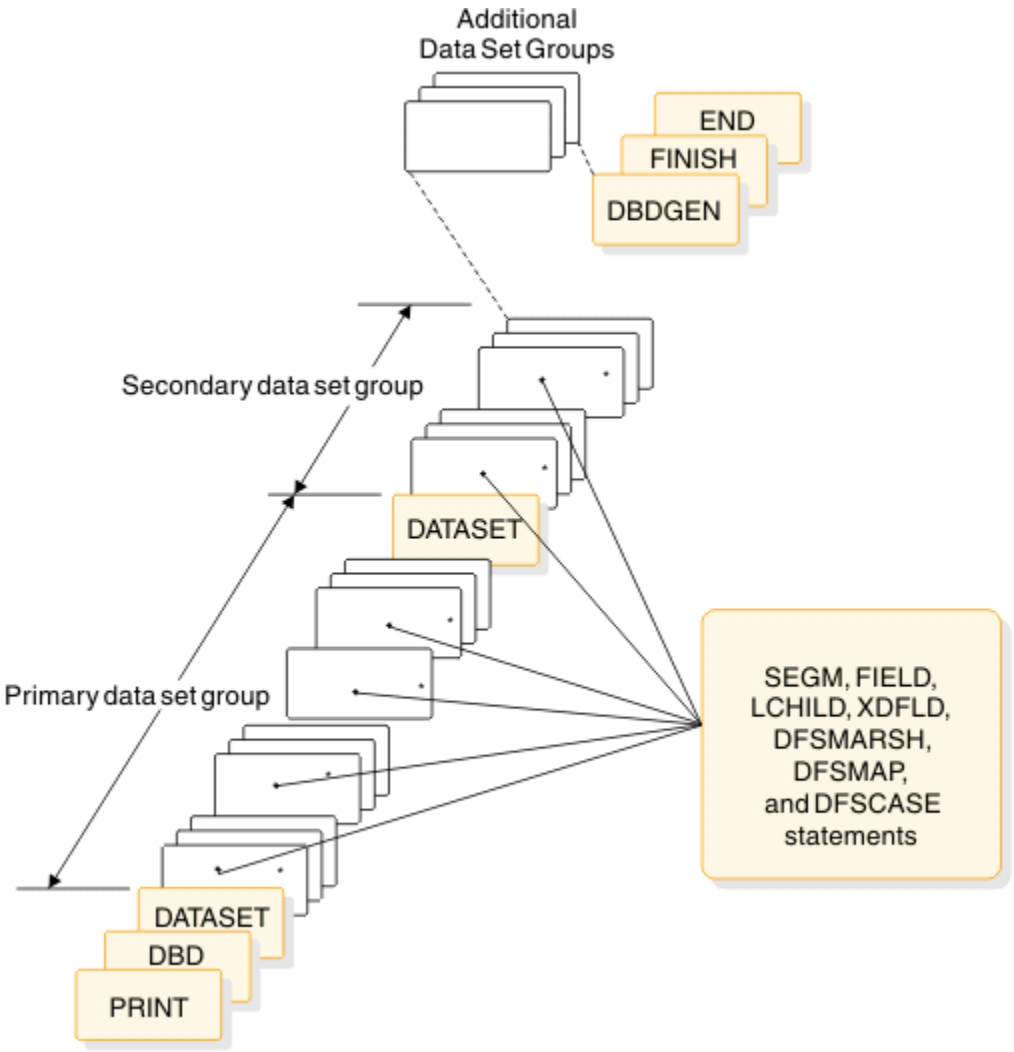


Figure 2. DBDGEN input record structure (except DEDB)

Exception: This input record structure applies to all DBDs except DEDB DBDs.

The PRINT statement is optional. If included, it is the first statement in the input deck. When PRINT is not included, the DBD control statement is first in the input deck. One or more DATASET statements follow the DBD statement. Each DATASET statement is followed by the SEGM, LCHILD, FIELD, XDFLD, DFSMARSH, DFSMAP, and DFSCASE statements that might be defined in that data set group. At least one SEGM statement must follow each DATASET statement. SEGM statements in the DBDGEN input set of records must be placed in the same hierarchic order as the segments in the database being defined.

FIELD and LCHILD statements follow the SEGM statement to which they apply. When a FIELD statement defines a sequence field within a segment, it must precede any XDFLD statements or any other FIELD statements that follow a SEGM statement. LCHILD statements follow the SEGM that defines a logical parent, HIDAM and PHIDAM root, and index target and index pointer segment types. When you are defining a secondary index relationship, the LCHILD statement that establishes the relationship must be followed by its corresponding XDFLD statements. No unrelated LCHILD statements can intervene between the two. XDFLD statements follow a SEGM that defines an index target segment type for a secondary index. A separate input set of records is required for each database.

If a DFSMARSH statement is used to define additional metadata for a field, the DFSMARSH statement must follow the corresponding FIELD statement. IMS associates the DFSMARSH statement with the last FIELD statement to precede the DFSMARSH statement in the input.

If DFSMAP and DFSCASE statements are used to define alternative field mappings with in a segment, the FIELD statement referenced by the DEPENDSON parameter in the DFSMAP statement must precede the DFSMAP statement in the input.

Requirement: The DBDGEN statement is required.

If FINISH is used, it precedes the END statement. END is the last statement in the input record structure.

DEDB DBD generation input record structure

The input record set structure for a DEDB DBD generation is essentially the same as for the other types of DBD generation except that AREA statements are used instead of DATASET statements.

All AREA statements must immediately follow the DBD statement. The SEGM statements and their associated FIELD statements follow the last AREA statement in hierarchic order. SEGM statements must also be placed in the same hierarchic order as the segments in the database being defined.

For DEDB DBD generation:

- The data set group concept does not apply.
- A secondary index is permitted.
- Logical relationships between databases are not permitted.
- LCHILD and XDFLD statements are permitted.
- Sequential dependent segments cannot have dependents.
- A separate input set of records is required for each database.

The following figure shows the rules for structuring a DEDB DBD generation input set of records.

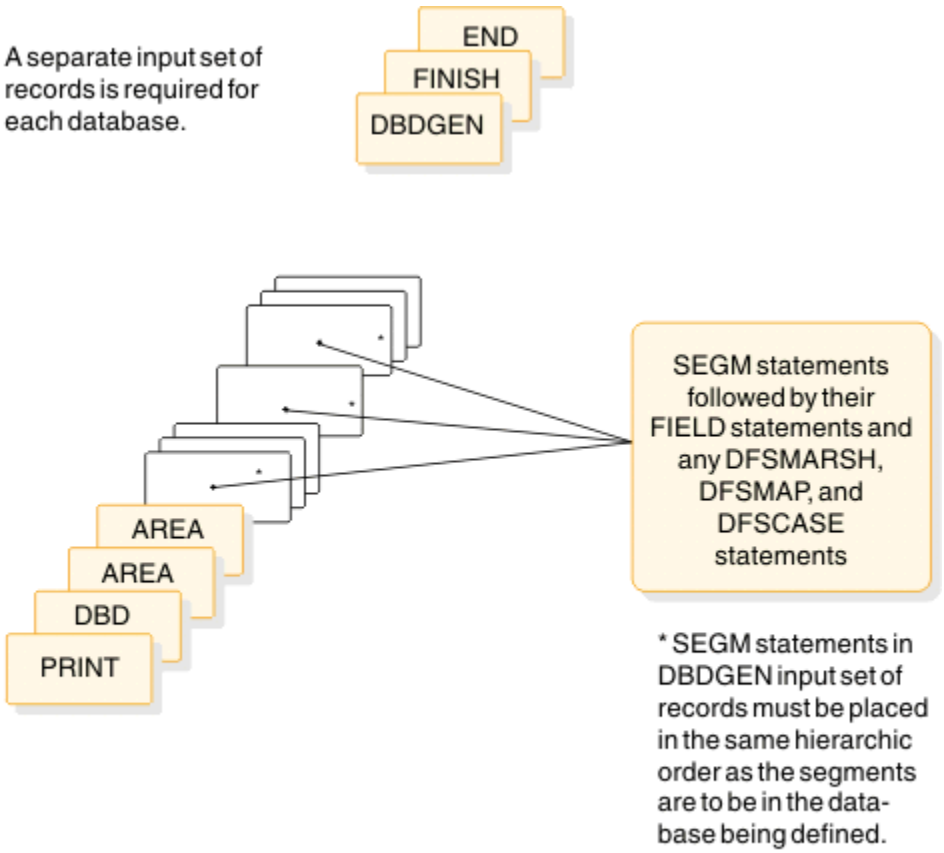


Figure 3. DEDB DBDGEN input record structure

DBD generation coding conventions

DBD generation statements are assembler language macro instructions. Each control statement must be identified by an operation code, for example: record-type code

Related concepts

Writing MVS macro instructions

DBDGEN statements

The DBDGEN utility uses statement instruction types as input to define a database.

The following list describes the macro statements that the DBDGEN utility accepts:

PRINT
Controls printing of assembly listing if present. This statement is optional.

DBD
Defines database name. This statement is required for all database types.

All full-function database names and DEDB database names must be unique.

DATASET
Defines a data set group within a database.

AREA
Defines an area within a Fast Path DEDB database.

All DEDB area names must be unique.

SEGM
Defines a segment type within a data set group or area.

LCHILD
Defines a logical or index relationship between segment types.

FIELD
Defines a field within a segment type.

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

XDFLD
Defines fields used with secondary indexes

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

DFSMARSH
Defines marshalling attributes for a field.

DFSMAP
For defining alternative field mappings in a segment, DFSMAP statements associate DFSCASE statements with the control field in the segment that identifies the particular DFSCASE statement in effect in segment instance. This statement is required only when a segment uses alternative field mappings.

DFSCASE
Defines a map case for a segment type that uses alternative field mapping. This statement is required only when a segment uses alternative field mappings.

DBDGEN
The DBDGEN statement indicates the end of DBD generation statements used to define the DBD. This statement is required.

FINISH
The FINISH statement is optional and is retained for compatibility.

END
The END statement indicates the end of input statements to the assembler. This statement is required.

The following table shows the statement instruction types that are used as input to the DBDGEN utility to define a database. Also included is the general use of each type of statement and the number of each type used per DBD generation.

The set of DBDGEN statements that each database type requires can differ. In the table, the numbers shown for each statement indicate whether the statement is required, optional, or does not apply for each database type.

Table 1. DBD generation statement instruction summary

Macro	Number used per DBD generation										
	HSAM/ SHSAM	**GSAM**	**HISAM/ HDAM**	**PHDAM**	**HIDAM**	**PHIDAM**	**MSDB**	**DEDB**	**Index**	**PSINDEX**	**Logical**
PRINT	0-1	0-1	0-1	0-1	0-1	0-1	0-1	0-1	0-1	0-1	0-1
DBD	1	1	1	1	1	1	1	1	1	1	1
DATASET	1	1	1/1-10	N/A	1-10	N/A	1	0	1	N/A	1
AREA	0	0	0	0	0	0	0	1-2048	0	0	
SEGM	1-255[3]	0-1	1-255	1-255	1-255	1-255	1	1-127	1[1]	1[1]	1-255
LCHILD	0	0	0-255	0-255	1-255	1-255	0	0	1[1]	1[1]	0
FIELD	0-1000	0-1000	1-1000	0-1000	1-1000	1-1000	0-255	1-1000	1[2]	1[2]	0
XDFLD	0	0	0-1000	0-1000	0-1000	0-1000	0	0-1000	0	0	0
DFSMARSH	0-1000	0-1,000	0-1000	0-1000	0-1000	0-1000	0-255	0-1000	1[2]	1[2]	0
DFSMAP	0 or more. No set limits	0 or more. No set limits.	0 or more. No set limits	0 or more. No set limits	0 or more. No set limits	0 or more. No set limits	0 or more. No set limits	0 or more. No set limits	0 or more. No set limits	0 or more. No set limits	0 or more. No set limits
DFSCASE	0 or more. No set limits	0 or more. No set limits.	0 or more. No set limits	0 or more. No set limits	0 or more. No set limits	0 or more. No set limits	0 or more. No set limits	0 or more. No set limits	0 or more. No set limits	0 or more. No set limits	0 or more. No set limits
DBDGEN	1	1	1	1	1	1	1	1	1	1	1
FINISH	1	1	1	1	1	1	1	1	1	1	1
END	1	1	1	1	1	1	1	1	1	1	1

Notes:

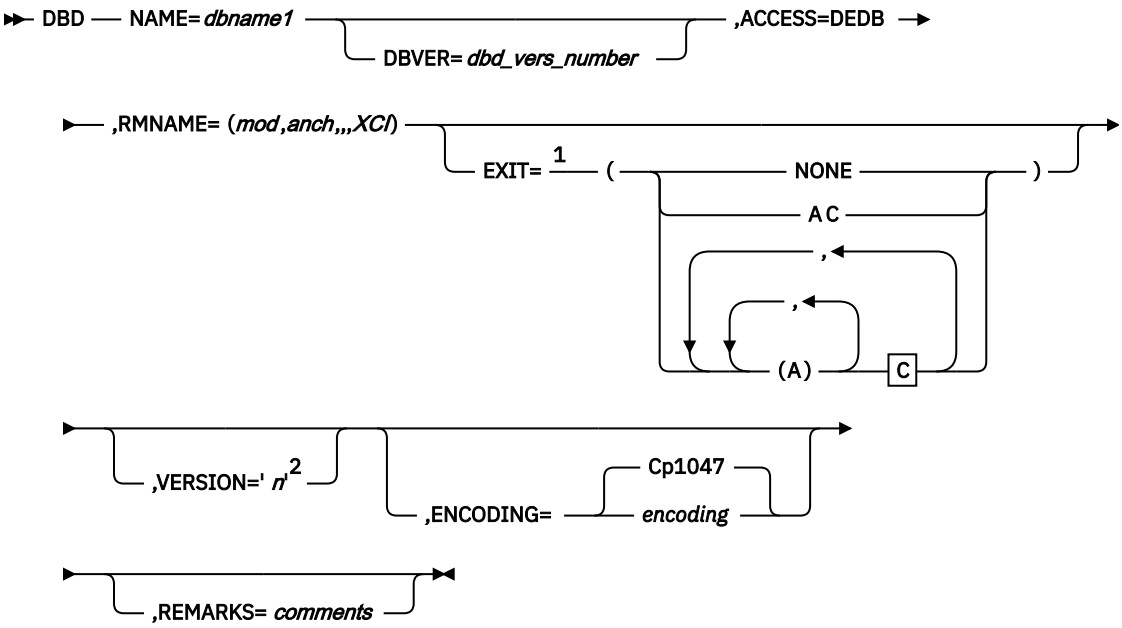
1. Maximum of 16 for a secondary index database.
2. Maximum of 1000 for a secondary index database.
3. A SHSAM database can only have one SEGM statement.

DBD statements

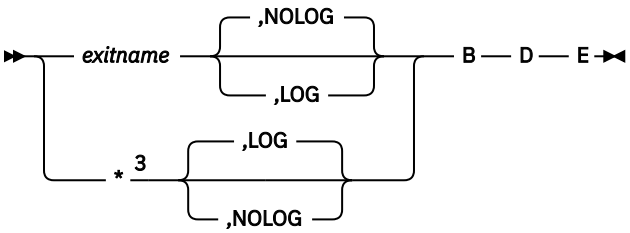
The DBD statement names the database being described and provides DL/I with information concerning its organization. There can be only one DBD control statement in the control statement input deck.

The format of the DBD macro instruction for each database type is shown in the following examples.

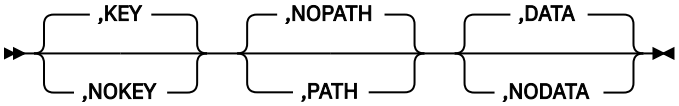
DEDB database DBD statement


DBD
NAME=dbname1
DBVER=dbd_vers_number
,ACCESS=DEDB
,RMNAME= (mod,anch,,,XCI)
EXIT=
1
(
NONE
)
A
C
,
,
(A)
C
,VERSION='n'
2
,ENCODING=
Cp1047
encoding
,REMARKS= comments

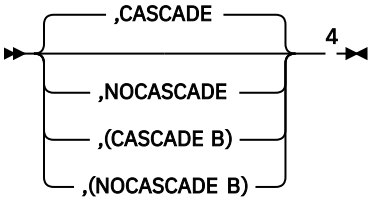
A


exitname
,NOLOG
,LOG
B
D
E
*
3
,LOG
,NOLOG

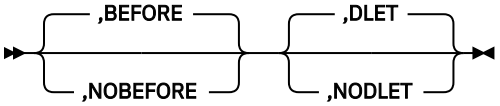
B


,KEY
,NOKEY
,NOPATH
,PATH
,DATA
,NODATA

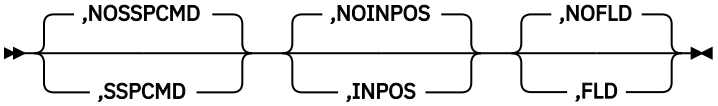
C


,CASCADE
,NOCASCADE
,(CASCADE B)
,(NOCASCADE B)
4

D


,BEFORE
,NOBEFORE
,DLET
,NODLET

E


,NOSSPCMD
,SSPCMD
,NOINPOS
,INPOS
,NOFLD
,FLD

Notes:

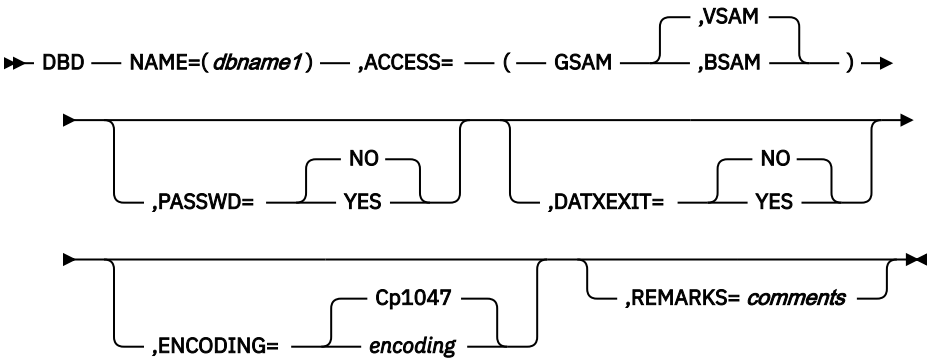
[1] Used for the Data Capture exit routine. You can specify more than one exit routine on a DBD statement.

[2] The default is an automatic DBDGEN time stamp.

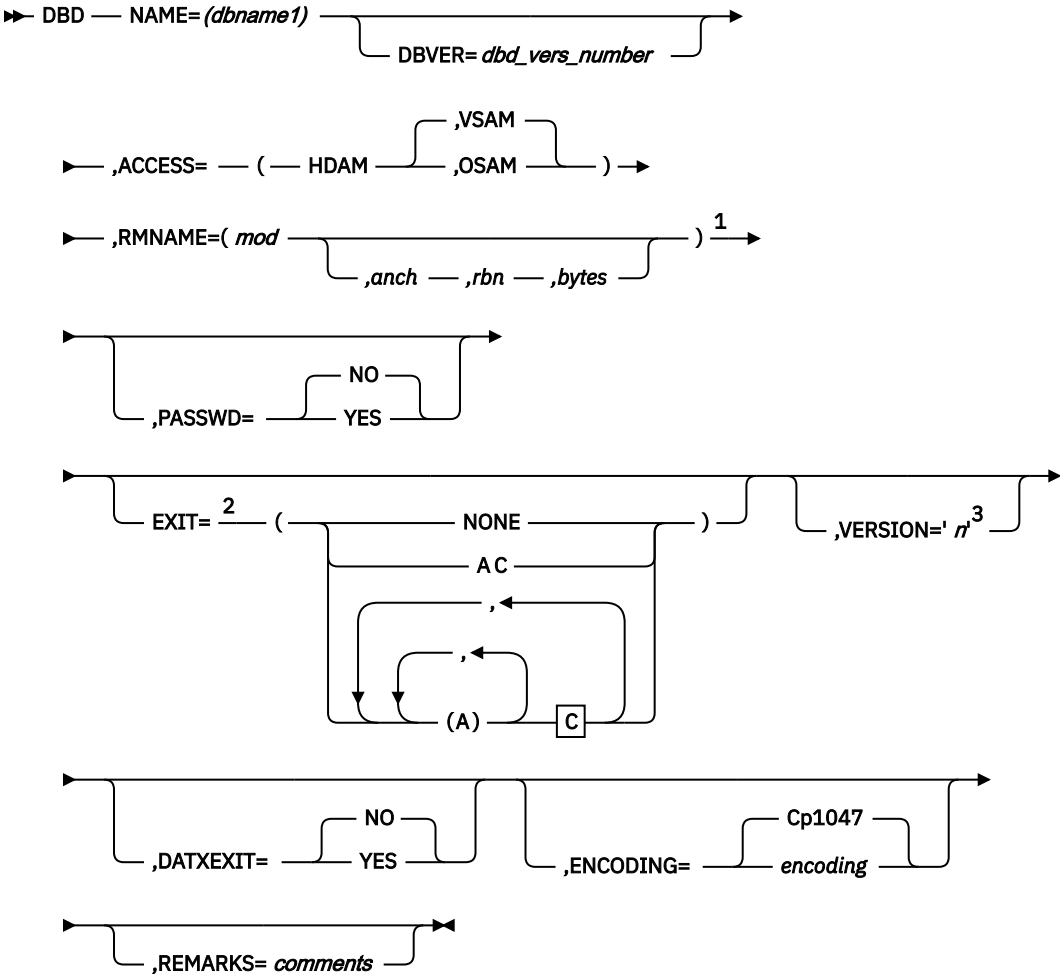
[3] If an exit routine name is not required because only logging is requested, specify the exit name as an asterisk (*). The default logging parameter in this case is LOG. If you specify an exit routine name, the default logging parameter is NOLOG.

[4] Used to control the CASCADE options. If any (CASCADE B) suboptions are specified with NOCASCADE in an **EXIT=** parameter, they will be ignored, and the value for those suboptions will be set to NO*xxxx*.

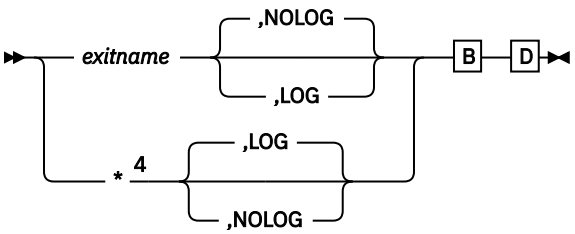
GSAM database DBD statement



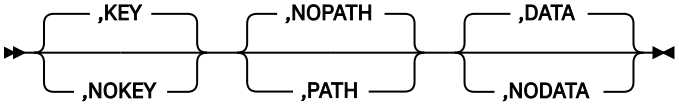
HDAM database DBD statement



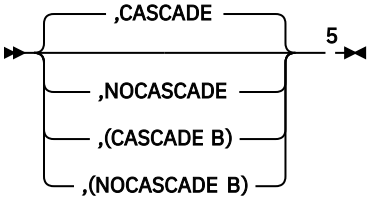
A



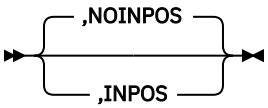
B



C



D



Notes:

[1] Optional operands, such as anch and rbn, might be required by certain randomizing modules. See the documentation for the randomizing module you are using.

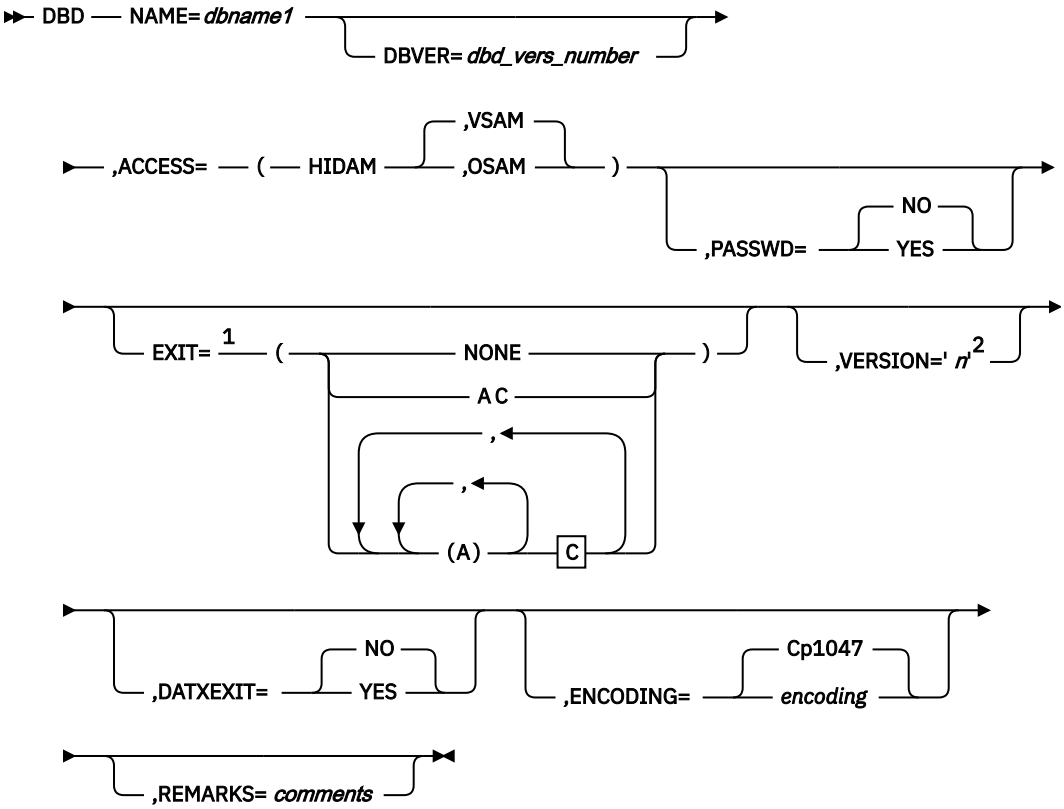
[2] Used for the Data Capture exit routine. You can specify more than one exit routine on a DBD statement.

[3] The default is an automatic DBDGEN time stamp.

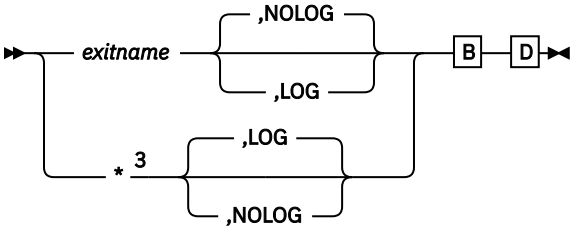
[4] If an exit routine name is not required because only logging is requested, specify the exit name as an asterisk (*). The default logging parameter in this case is LOG. If you specify an exit routine name, the default logging parameter is NOLOG.

[5] Used to control the CASCADE options. If any (CASCADE B) suboptions are specified with NOCASCADE in an **EXIT=** parameter, they will be ignored, and the value for those suboptions will be set to NO*xxxx*.

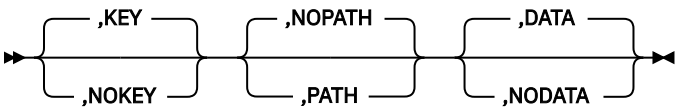
HIDAM database DBD statement



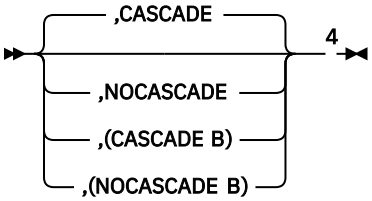
A



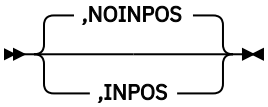
B



C



D



Notes:

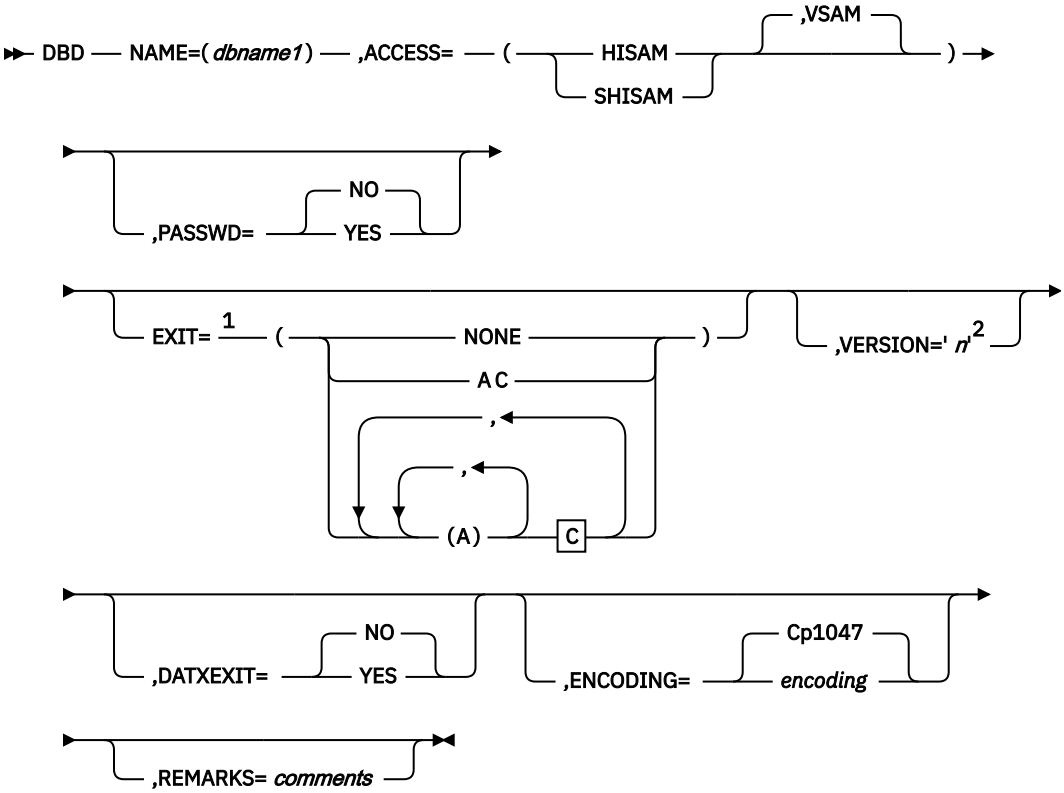
[1] Used for the Data Capture exit routine. You can specify more than one exit routine on a DBD statement.

[2] The default is an automatic DBDGEN time stamp.

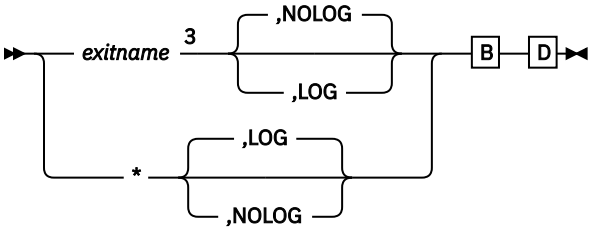
[3] If an exit routine name is not required because only logging is requested, specify the exit name as an asterisk (*). The default logging parameter in this case is LOG. If you specify an exit routine name, the default logging parameter is NOLOG.

[4] Used to control the CASCADE options. If any (CASCADE B) suboptions are specified with NOCASCADE in an **EXIT=** parameter, they will be ignored, and the value for those suboptions will be set to NO*xxxx*.

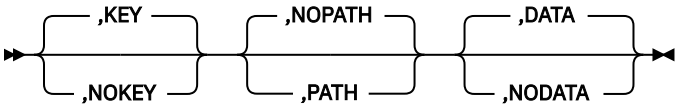
HISAM/SHISAM database DBD statement



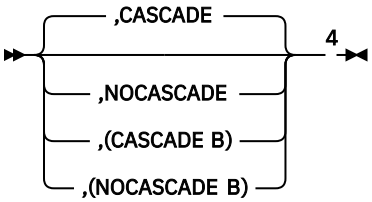
A



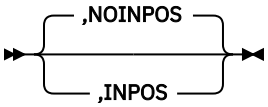
B



C



D



Notes:

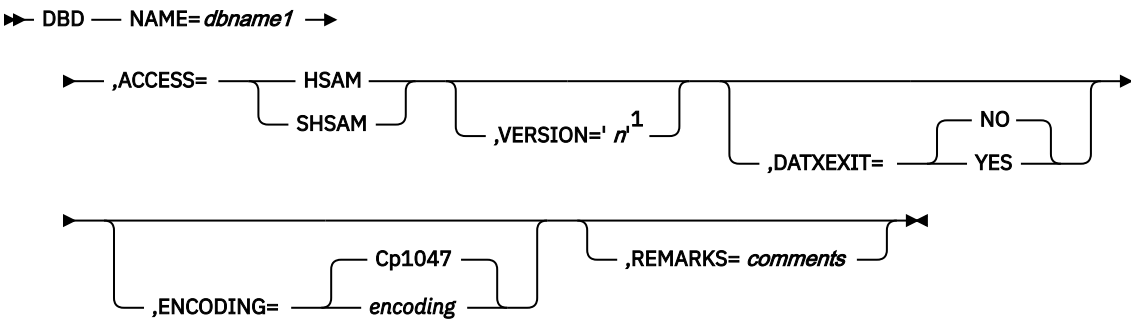
[1] Used for the Data Capture exit routine. You can specify more than one exit routine on a DBD statement.

[2] The default is an automatic DBDGEN time stamp.

[3] If an exit routine name is not required because only logging is requested, specify the exit name as an asterisk (*). The default logging parameter in this case is LOG. If you specify an exit routine name, the default logging parameter is NOLOG.

[4] Used to control the CASCADE options. If any (CASCADE B) suboptions are specified with NOCASCADE in an **EXIT=** parameter, they will be ignored, and the value for those suboptions will be set to NO*xxxx*.

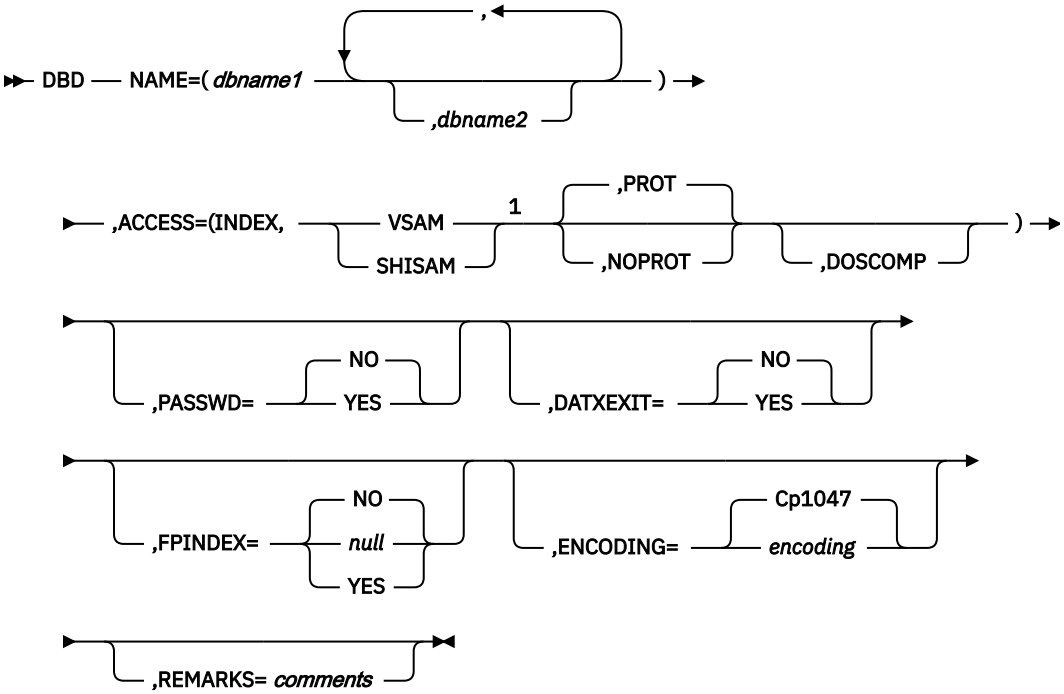
HSAM/SHSAM database DBD statement



Notes:

[1] The default is an automatic DBDGEN time stamp.

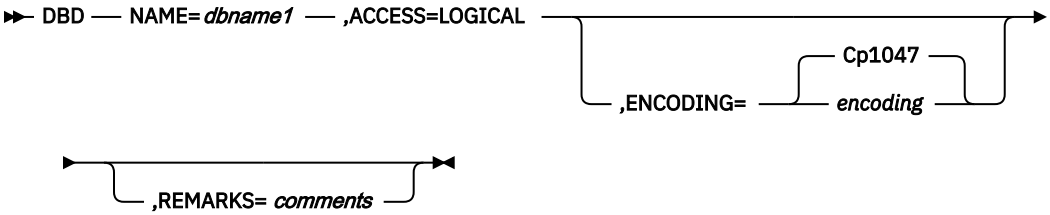
INDEX database DBD statement



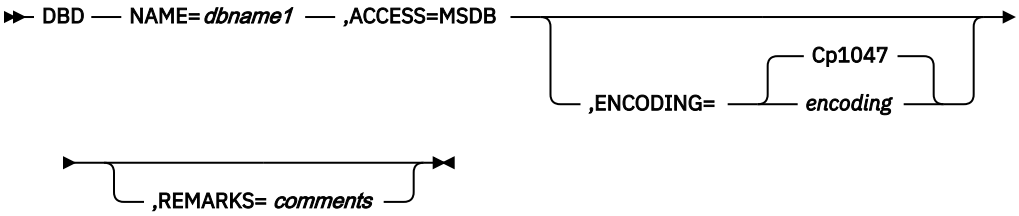
Notes:

[1] A full-function secondary index must use VSAM. A Fast Path secondary index can use either VSAM or HISAM.

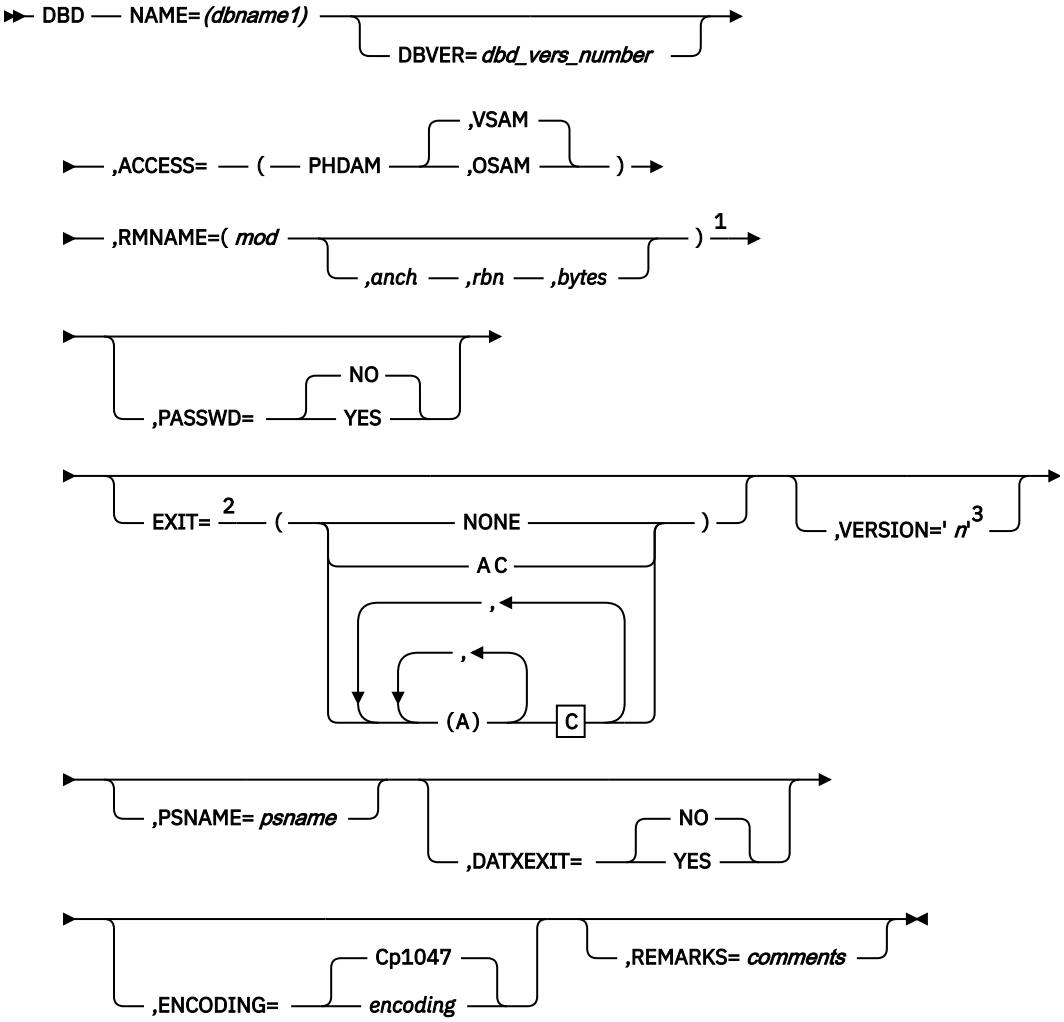
LOGICAL database DBD statement



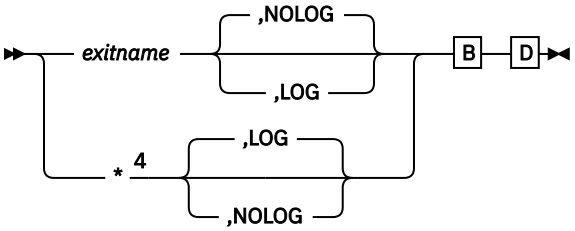
MSDB database DBD statement



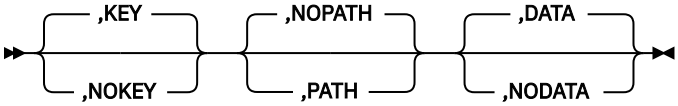
PHDAM database DBD statement



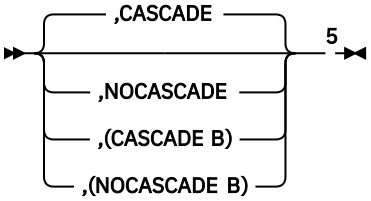
A



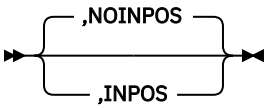
B



C



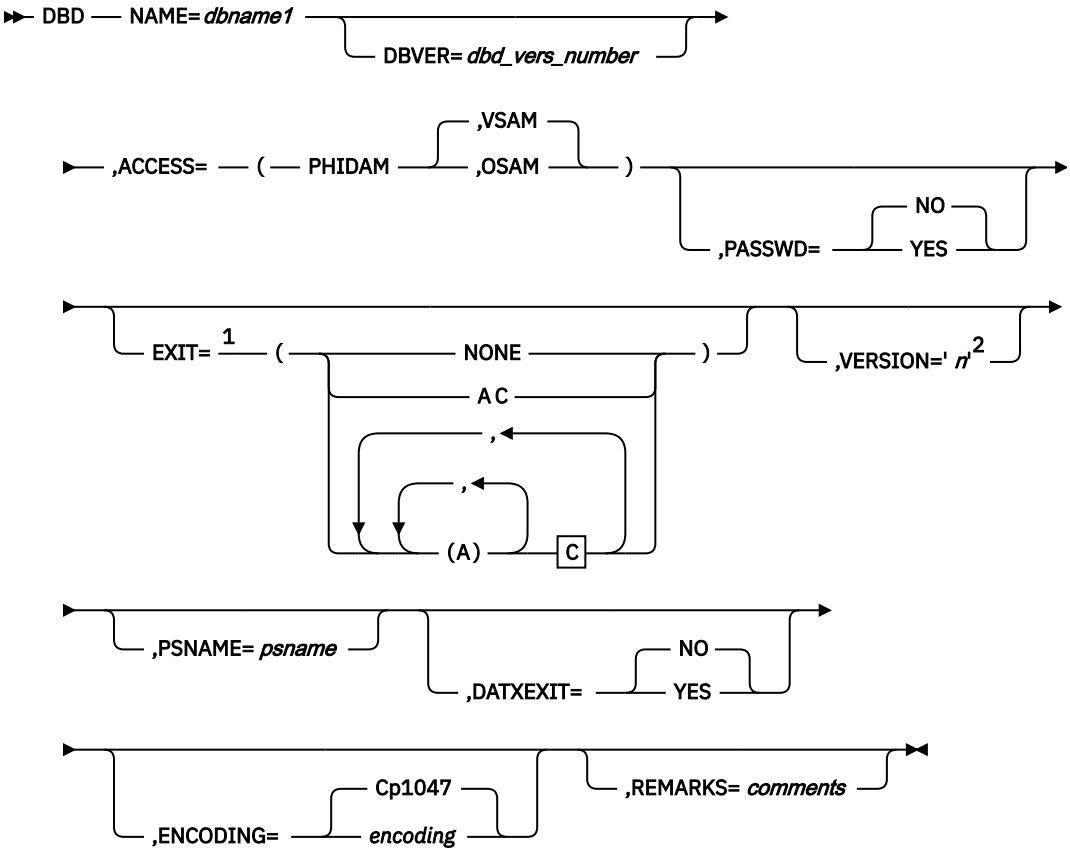
D



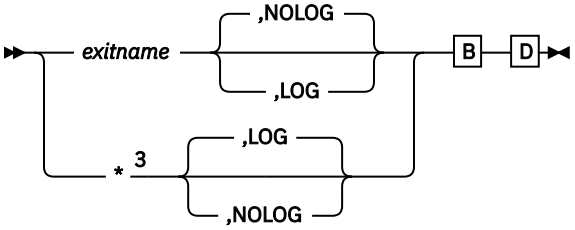
Notes:

[1] Optional operands, such as anch and rbn, might be required by certain randomizing modules. See the documentation for the randomizing module you are using.
[2] Used for the Data Capture exit routine. You can specify more than one exit routine on a DBD statement.
[3] The default is an automatic DBDGEN time stamp.
[4] If an exit routine name is not required because only logging is requested, specify the exit name as an asterisk (*). The default logging parameter in this case is LOG. If you specify an exit routine name, the default logging parameter is NOLOG.
[5] Used to control the CASCADE options. If any (CASCADE B) suboptions are specified with NOCASCADE in an **EXIT=** parameter, they will be ignored, and the value for those suboptions will be set to NO*xxxx*.

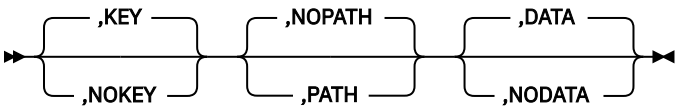
PHIDAM database DBD statement



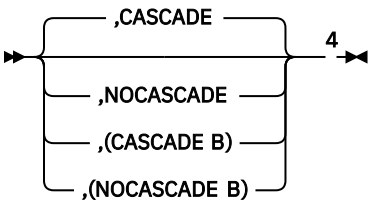
A



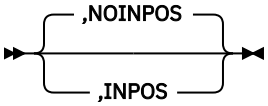
B



C



D



Notes:

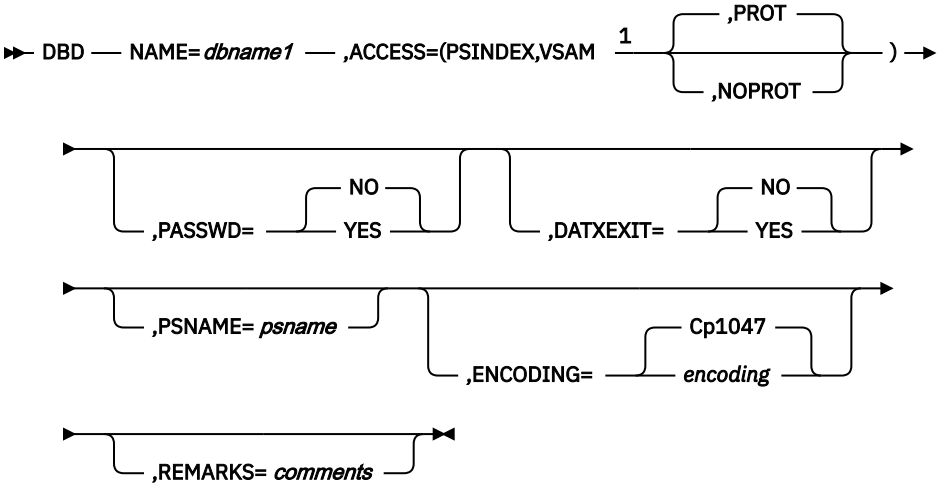
[1] Used for the Data Capture exit routine. You can specify more than one exit routine on a DBD statement.

[2] The default is an automatic DBDGEN time stamp.

[3] If an exit routine name is not required because only logging is requested, specify the exit name as an asterisk (*). The default logging parameter in this case is LOG. If you specify an exit routine name, the default logging parameter is NOLOG.

[4] Used to control the CASCADE options. If any (CASCADE B) suboptions are specified with NOCASCADE in an **EXIT=** parameter, they will be ignored, and the value for those suboptions will be set to NO*xxxx*.

PSINDEX database DBD statement



Notes:

[1] A secondary index must use VSAM.

DBD statement parameter descriptions

DBD
Identifies this statement as the DBD control statement.

NAME=

Specifies the name of the database being described. The name can be from 1 to 8 alphanumeric characters. Do not give a database the same name as an existing PSB or program view. Using an existing name can cause unpredictable results.

If an existing name is used, an error occurs during ACB generation.

This name can be the same as that specified in the DD1 parameter of the first DATASET control statement.

For a shared secondary index database, the names of up to 16 secondary index DBDs can be specified.

DBVER=
A numeric value from 1 to 2147483647 that identifies a specific version of a DBD when multiple DBDs are used by application programs to access the same database.

The value specified on the DBVER keyword must be one greater than the highest DBVER value of any prior version of the DBD that is stored in the database record in the IMS catalog.

The DBVER keyword is optional. If the DBVER keyword is omitted, the version of the DBD is 0, even if database versioning is not enabled.

The DBVER keyword is valid only for the following database types:

- DEDB
- HDAM
- HIDAM
- PHDAM
- PHIDAM

ACCESS=

Specifies the DL/I access method and the operating system access method to be used for this database. This keyword also defines the secondary index database as a HALDB. The different access methods are:

HSAM
Hierarchical sequential access method (HSAM). When HSAM is specified, and only one segment type is defined in the HSAM database, this parameter defaults to SHSAM.

SHSAM
Simple HSAM database that contains only one fixed-length segment type. When a simple HSAM database is defined, no prefix is required in occurrences of the segment type to enable IMS to process the database.

GSAM
Generalized sequential access method (GSAM). BSAM or VSAM can be specified as the operating system access method. VSAM is the default. When GSAM is specified, no SEGM control statement is allowed in the DBD.

HISAM
Hierarchical index sequential access method (HISAM). VSAM can be specified as the operating system access method. It is the default.

SHISAM
Simple HISAM database that contains only one fixed-length segment type. A simple HISAM database can only be specified when VSAM is specified as the operating system access method. When a simple HISAM database is defined, no prefix is required in occurrences of the segment type to enable IMS to process the database.

HDAM
Hierarchical direct access method (HDAM). OSAM or VSAM can be specified as the operating system access method. VSAM is the default.

PHDAM
Partitioned hierarchical direct access method (PHDAM). OSAM or VSAM can be specified as the operating system access method. VSAM is the default.

HIDAM
Hierarchical indexed direct access method (HIDAM). OSAM or VSAM can be specified as the operating system access method. VSAM is the default.

PHIDAM
Partitioned hierarchical indexed direct access method (PHIDAM). OSAM or VSAM can be specified as the operating system access method. VSAM is the default.

MSDB
Main storage database (MSDB).

DEDB
Data entry database (DEDB).

INDEX
Creates the primary index to occurrences of the root segment type in a HIDAM database, or creates a secondary index to a segment type in a HISAM, HDAM, or HIDAM database. For the primary or secondary index to a HIDAM database, VSAM must be specified as the operating system access method.

The INDEX parameter is also used to create a secondary index for a DEDB database. In such a case, VSAM and SHISAM are both valid operating system access types.

The INDEX parameter is not used to define the primary index of a PHIDAM database.

PSINDEX
Creates the partitioned secondary index to a segment type in PHDAM and PHIDAM databases. VSAM must be specified as the operating system access method. VSAM is the default.

LOGICAL
A logical database comprises logical concatenations of some or all of one or more physical databases. Logical databases must reference existing physical databases.

PROT | NOPROT

Specifies if a secondary index database uses index pointer protection. This optional parameter ensures the integrity of all fields in index pointer segments that are used by IMS. Use of this parameter prevents an application program from doing a replace operation on any field within an index pointer segment except for fields within the user data portion of index pointer segments. Delete operations are still enabled for index pointer segments. If a delete is issued for an index pointer segment, the index target segment pointer in the index pointer segment is deleted. However, the index source segment that caused the index pointer segment to be created originally is not deleted.

If index pointer protection is not used, an application program can replace all fields within an index pointer segment except the constant, search, and subsequence fields. Inserts to an index database are invalid under all conditions.

By default, a secondary index database uses index pointer protection.

DOSCOMP
Indicates if this is a DLI/DOS index database. Must be specified if the database is an index, and it was created using DLI/DOS. DLI/DOS index databases contain a segment code as part of the prefix. Specifying that a database is a DLI/DOS index database causes IMS to expect this code to be present in the defined database, and to process in a way that preserves this code. This includes providing a segment code for new segments being inserted. DLI/DOS databases must use VSAM and cannot be PHDAM, PHIDAM, or PSINDEX databases.

PSNAME=
Specifies the module that selects the HALDB partition for PSINDEX, PHDAM, or PHIDAM databases. The parameter is a HALDB partition selection exit routine module name. This parameter is only valid when the access type for the database is PSINDEX, PHDAM, or PHIDAM.

Exception: A user-provided HALDB partition selection routine is not needed if root key ranges define HALDB partition membership.

RMNAME=

Specifies a module name that is used to manage the data that is stored in a DEDB or in the primary data set group of an HDAM or PHDAM database. This parameter is only valid when the database access type is HDAM, PHDAM, or DEDB. A randomizing module controls root segment placement in or retrieval from the DEDB, HDAM, or PHDAM database. One or more modules, called randomizing modules, can be utilized within the IMS system. A particular database has only one randomizing module associated with it. A generalized module, which uses user-supplied parameters to perform randomizing for a particular database, can be written to service several databases. The purpose of a randomizing module is to convert a value supplied by an application program for root segment placement in, or retrieval from, a DEDB, HDAM, or PHDAM database into a relative block number and anchor point number. You can randomize within an area by selecting a two-stage randomizer. When you select a two-stage randomizer, the number of root anchor points in an area can be changed without having to stop all areas in the DEDB with the **/DBRECOVERY** command.

For PHDAM databases, the randomizer module names and values become the default for each partition. You can set a different randomizer name and values for each partition during HALDB partition definition. HALDB partition selection is done prior to invoking the randomizing module. The randomizing module selects locations only within a partition.

mod
The module name is the 1- to 8-character alphanumeric name of a user-supplied randomizing module that is used to store and access segments in this DEDB, PHDAM, or HDAM database. Select a two-stage randomizer by specifying the randomizer name in the module name parameter and 2 in the anchor point parameter.

anch
The purpose of the anch value is different depending on whether you are defining a Fast Path DEDB database or a full-function HDAM or PHDAM database.

This parameter must be an unsigned decimal integer. The default value of this parameter is one.

For a DEDB database, the value of *anch* specifies the type of randomizer. A value of 1 indicates a single-stage randomizer. A value of 2 indicates a two-stage randomizer. Any other value is invalid.

For HDAM and PHDAM databases, the value of *anch* specifies the number of root anchor points desired in each control interval or block in the root addressable area of the HDAM or PHDAM database. Typical values are from 1 to 5 and the value cannot exceed 255.

When accessing a HDAM or PHDAM database, if a user randomizing routine produces an anchor point number greater than the number specified for this parameter, the highest-numbered anchor point in the control interval or block is used. When a randomizing routine produces an IMS anchor point number of zero, IMS uses anchor point one in the control interval or block.

rbn
Specifies the maximum relative block number value that you want to allow a randomizing module to produce for this database. This parameter is for HDAM or PHDAM databases only. This value

determines the number of control intervals or blocks in the root addressable area of an HDAM or PHDAM database. This parameter must be an unsigned decimal integer whose value does not exceed 2^{24}-1. If this parameter is omitted, no upper limit check is performed on the relative block number created by the randomizing module. If this parameter is specified, but the specified randomizing module produces an relative block number greater than this parameter, the highest control interval or block in the root addressable area is used by IMS. If a user randomizing module produces a block number of zero, the control interval or block one is used by IMS.

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

bytes
Specifies the maximum number of bytes of database record that can be stored into the root addressable area in a series of inserts unbroken by a call to another database record. This parameter is for HDAM and PHDAM databases only. If this parameter is omitted, no limit is placed on the maximum number of bytes of a database record that can be inserted into this database's root segment addressable area. The bytes parameter must be an unsigned decimal integer whose value does not exceed 2^{24}-1. When the maximum relative block number parameter is omitted, this parameter is ignored. In this case, there is no limit on the number of bytes of a database record that can be inserted into the root addressable area.

If this parameter is specified for an HDAM or PHDAM database and the length of the database record is larger, the remainder of the record is inserted into the overflow area following the current end-of-file (EOF). This operation requires that enough space be available after the current EOF to contain the remainder of all database records that exceed the value of this parameter. If sufficient space is not available in the overflow area following the current EOF, the database records are inserted randomly in the database.

XCI
Specifies whether this DEDB uses the Extended Call Interface when making calls to the randomizer. This option allows the randomizer to be called in three different ways. On initialization of IMS or during a **/START DB** command, IMS will first load the randomizer and then make an INIT call to the randomizer to invoke its initialization routines. During a **/DBR DB** command, IMS will make a TERM call to the randomizer to invoke the termination routines before unloading the randomizer. The normal randomizing call to the randomizer is made when the application issues a GU or ISRT call on a root segment. The XCI option is only valid for DEDBs.

PASSWD=
Prevents accidental access of IMS databases by non-IMS programs.

YES
Specifying PASSWDYES causes DL/I to use the database name as the VSAM password when opening any data set for this database. This parameter is only valid for databases that use VSAM as the access method. You cannot use the database name as the password for the LOGICAL or DEDB database types. When the user defines the VSAM data sets for this database using the DEFINE statement of z/OS Access Method Services, the control level (CONTROLPW) or master level (MASTERPW) password must be the same as the DBDNAME for this DBD. All data sets associated with this DBD must use the same password.

For the IMS DB/DC system, all VSAM OPENs bypass password checking and thus avoid operator password prompting. For the IMS DB system, VSAM password checking is performed. In the batch environment, operator password prompting occurs if automatic password protection is not specified, and the data set is password protected at the control level (CONTROLPW) with passwords not equal to the database name.

NO
Specifying PASSWDNO indicates that the database name should not be used as the VSAM password. This is the default behavior.

EXIT=

Specifies that the Data Capture exit routine is used. You can specify multiple exit routine names on a single DBD statement. You can select different data options for each exit routine. The order that you list the exit routines within the parameter determines the order the exit routines are called for the segment.

When specified on the DBD statement, the EXIT= parameter applies to all segments within the physical database that do not have the EXIT= parameter on the SEGM statement. The following physical databases are supported by this exit routine:

- HISAM
- SHISAM
- HDAM
- PHDAM
- HIDAM
- PHIDAM
- DEDB

If the exit routine is not specified for a supported database organization or a supported segment type, DBDGEN fails.

If the job name of a CCTL or ODBM address space is specified on the **SUPPDCAPNAME=** parameter, which is in the DATABASE section of the DFSDF*xxx* member of the IMS PROCLIB data set, the exit routine is not called for data updates invoked by the specified job, even if a Data Capture exit routine is specified on the EXIT= parameter.

The EXIT= parameter can also be specified on the SEGM statement.

exit_name

Specifies the name of the exit routine that processes the data. This parameter is required. The name must follow standard naming conventions. A maximum of 8 alphanumeric characters is allowed. You can specify an asterisk (*) instead of an exit routine name to indicate that you want logging only. If this is done, the logging parameter default is LOG. If you do specify an exit routine, the logging parameter default is NOLOG. All of the following operands are optional.

KEY | NOKEY

Specifies whether the exit routine is passed the physical concatenated key. This key identifies the physical segment that is updated by the application.

KEY

Specifies the exit routine is passed the physical concatenated key.

KEY is the default.

NOKEY

Can be specified when the physical concatenated key is not required for the exit routine.

DATA | NODATA

Specifies whether the physical segment data is passed to the exit routine for updating.

DATA

Specifies that the physical segment data is passed to the exit routine for updating. When DATA is specified and a Segment Edit/Compression exit routine is also used, the data passed is expanded data.

DATA is the default.

NODATA

Can be specified when the exit routine does not require segment data. Use NODATA to avoid the overhead that is created from saving physical segment data.

NOPATH | PATH

Specifies whether the exit routine requires data from segments in the physical root's hierarchical path.

NOPATH
Indicates that the exit routine does not require data from segments in the physical root's hierarchical path. NOPATH is an efficient way to avoid the processing time that is needed to retrieve path data.

NOPATH is the default.

PATH
Can be specified when the data from each segment in the physical root's hierarchical path must be passed to the exit routine for an updated segment. Use PATH to allow an application to separately access several segments for insertion, replacement, or deletion.

You can use the PATH option when information from segments in the path is needed to compose the Db2 for z/OS primary key. The Db2 for z/OS primary key would then be used in a propagation request for a dependent segment update. Typically, you need this kind of segment information when the parent contains the key information and the dependent contains additional data that would not fit in the parent segment.

You can also use PATH when additional processing is necessary. It could be that you are not accessing several segments with one call; for example, you did not invoke the D command code. In this case, additional processing is necessary if the application is to access each segment with a separate call.

DLET | NODLET

Specifies whether X'99' log records are written for DLET calls.

Note: DLET or NODLET can only be specified for a DEDB.

If you specify this parameter in the SEGM statement, it overrides the specification for the DBD statement.

DLET
X'99' log records are written for DLET calls.

DLET is the default.

NODLET
No X'99' log records are written for DLET calls.

BEFORE | NOBEFORE

Specifies whether the before data is included in X'99' log records for REPL calls.

Note: BEFORE or NOBEFORE can only be specified for a DEDB.

If you specify this parameter in the SEGM statement, it overrides the specification for the DBD statement.

BEFORE
Before data is included in X'99' log records for REPL calls.

BEFORE is the default.

NOBEFORE
No before data is included in X'99' log records for REPL calls.

CASCADE | NOCASCADE
Specifies whether the exit routine is called when DL/I deletes this segment.

CASCADE
Indicates that the exit routine is called when DL/I deletes this segment because the application deleted a parent segment. Using CASCADE ensures that data is captured for the defined segment.

CASCADE is the default.

The CASCADE parameter has three suboptions. These suboptions control the way data is passed to the exit routine. If you specify suboptions, you must enclose the CASCADE parameter and the suboptions within parentheses.

KEY | NOKEY
Specifies whether the physical concatenated key is passed to the exit.

KEY
Passes the physical concatenated key to the exit. This key identifies the segment being deleted by a cascade delete.

KEY is the default.

NOKEY
Can be used when the exit routine does not require the physical concatenated key of the segment being deleted.

DATA | NODATA
Specifies whether segment data is passed to the exit routine for a cascade delete.

DATA
Passes segment data to the exit routine for a cascade delete. DATA also identifies the segment being deleted when the physical concatenated key is unable to do so.

DATA is the default.

NODATA
Can be specified when the exit routine does not require segment data. NODATA reduces the significant storage and performance requirements that result from saving physical segment data.

NOPATH | PATH
Specifies whether the exit routine requires segment data in the physical root's hierarchical path.

NOPATH
Indicates that the exit routine does not require segment data in the physical root's hierarchical path. Use NOPATH to eliminate the substantial amount of path data needed for a cascade delete.

NOPATH is the default.

PATH
Can be specified to allow an application to separately access several segments for a cascade delete.

NOCASCADE
Indicates that the exit routine is not called when DL/I deletes this segment. Cascade delete is not necessary when a segment without dependents is deleted.

Note: If any (CASCADE B) suboptions are specified with NOCASCADE in an **EXIT=** parameter, they will be ignored, and the value for those suboptions will be set to NO*xxxx*.

LOG | NOLOG
Specifies whether data capture control blocks or data is written to the IMS system log.

LOG
Requests that the data capture control blocks and data be written to the IMS system log.

For more information, see Asynchronous data propagation (System Programming APIs).

NOLOG
Indicates that no data capture control blocks or data is written to the IMS system log.

NOSSPCMD | SSPCMD
An optional parameter that indicates whether command codes related to Fast Path subset pointers (SSP) be captured. The default is NOSSPCMD. It is recommended that this option be specified only on segments that involve subset pointers.

The following table indicates which command codes are captured for a given DL/I call:

Table 2. Command codes that are captured for DL/I calls

DL/I call	Details
G* (get calls)	M, S, W, Z. R is captured if at least one of M, S, W, or Z is also on the same SSA, or along with the PATH data if PATH is requested.
REPL	M, S, W, Z
DLET	Z
ISRT	M, S, W, Z. R is captured if the segment is being inserted or if it was specified on an SSA of a segment not being inserted but PATH data is requested.

NOINPOS | INPOS
An optional parameter to request that the next twin data be captured on an ISRT call. The default is NOINPOS. The twin data of the twin that follows an inserted segment will be captured if INPOS is specified and the following conditions are true:

- An ISRT of a non-unique segment is made.
- An ISRT rule of HERE is used.

If the new segment is the only twin instance or last in the twin chain, no twin data will be captured.

NOFLD | FLD
An optional parameter to request that updates that are made by a DEDB FLD call be captured. This option is valid only for a DEDB. The information captured is logged only in the X'9904' log records if option LOG is specified. It is not passed to the Data Capture exit routine.

VERSION=
Specifies a *character string* used to identify the DBD. The exit routine is passed this character string so it can determine the DBD version used to update the database.

character string
The character-string length can be up to 255 bytes. There are no checks to ensure that the proper values have been inserted. Therefore, it is important that the variable-length character string be updated whenever the DBD changes.

If you do not specify a character string, a 13-character time stamp is generated by DBDGEN. It represents the date and time the DBDGEN was completed. Its format is:

```
MM/DD/YYHH.MM
```

Where:

MM
The month

DD
The day of the month

YY
The last two digits of the year

HH
The hour on a 24-hour clock

MM
The minutes

DATXEXIT=

Allows a user exit, DFSDBUX1, to be used by an application while processing this database. The default is NO.

Allows the Data Conversion user exit routine (DFSDBUX1) to be used by an application while it is processing this database. The default is DATXEXITNO.

If DATXEXITYES is specified, the user exit DFSDBUX1 is called at the beginning and at the end of each database call. If DFSDBUX1 is not loaded, IMODULE is called to load it.

If DATXEXITNO is specified, the DFSDBUX1 user exit routine can be called, provided DFSDBUX1 is located in the SDFSRESL. If DFSDBUX1 does not need to be called again for the database definition, X'FF' is returned in the SRCHFLAG field in the JCB, and DFSDLA00 dynamically marks the database definition as not requiring the exit. In this case, the user exit is not called again for that database definition for the duration of the IMS session, unless the DMB is purged from the DMB pool.

FPINDEX=

Specifies whether an index database is a secondary index for a primary Fast Path DEDB database. By default, an index database is not a secondary index.

Valid values are NO, *null*, and YES. NO and *null* are equivalent. The default is NO (or *null*).

ENCODING=

An optional 1- to 25-character field that specifies the default encoding of all character data in this database.

The default code page is Cp1047, which specifies EBCDIC encoding.

This value cannot contain the following characters:

- Single or double quotation marks
- Blanks
- Less than (<) and greater than (>) symbols
- Ampersands (&)

This value can be overridden in individual segments or fields.

REMARKS=

Optional user comments. A 1- to 256-character field.

If your comments contain special characters, such as commas or blank spaces, enclose the full comment string in single quotation marks.

The value specified on the REMARKS keyword cannot contain the following characters:

- Single quotation marks, except when they are used to enclose the full comment string. If a single quotation mark is entered before the end of the full comment string, the remainder of the comment string is truncated. The following examples show correct and incorrect usages of single quotation marks on the REMARKS keyword:

  CORRECT

  ```
  REMARKS='These remarks apply to the XYZ application'
  ```

  INCORRECT

  ```
  REMARKS='These remarks apply to the 'XYZ' application'
  ```

- Double quotation marks.
- Less than (<) symbols.
- Greater than (>) symbols.
- Ampersands (&).

Related concepts

Data Capture exit routines (Database Administration)

Related information
DBD770 (Messages and Codes)

DATASET statements

A DATASET statement defines a data set group within a database.

At least one DATASET statement is required for each DBD generation, except for HALDB, DEDB, and LOGICAL databases. HALDB databases use the DSGROUP parameter on the SEGM statement instead of DATASET statements to define data set groups. DEDB databases use AREA statements instead of DATASET statements to define data set groups.

The maximum number of DATASET statements used depends on the type of databases. Some databases can have only one data set group. Data Entry databases can have 1 to 2048 areas defined. HDAM and HIDAM databases can be divided into 1 to 10 data set groups subject to the rules in "Rules for dividing a database into multiple data set groups" on page 48.

In the DBDGEN input deck, a DATASET statement precedes the SEGM statements for all segments that are to be placed in that data set group. The first DATASET statement of a DBD generation defines the primary data set group. Subsequent DATASET statements define secondary data set groups.

Exception: The only exception to the order of precedence is when the LABEL field of a DATASET statement is used. Refer to "Use of the LABEL field" on page 48 for this exception.

Comments must not be added to a subsequent labeled DATASET macro that has no operands.

Rules for dividing a database into multiple data set groups

HDAM and HIDAM databases can be divided into a maximum of 10 data set groups according to the following restrictions. Each DATASET statement creates a separate data set group, except for the case explained in "Use of the LABEL field" on page 48. The first DATASET statement defines the primary data set group. Subsequent DATASET statements define secondary data set groups.

For HDAM or HIDAM databases, you can use DATASET statements to divide the database into multiple data set groups at any level of the database hierarchy; however, the following restriction must be met. A physical parent and its physical children must be connected by physical child or physical twin pointers, as opposed to hierarchic pointers, when they are in different data set groups, as shown in the following figure.

The connection between segment A (the root segment in the primary data set group), and segment B (a first level dependent in the secondary data set group) must be made using a physical child. The connection between segment C (a first level dependent in the primary data set group) and segment D (a second level dependent in the secondary data set group) must also be made using a physical child. The connection between multiple occurrences of segments B and D under one parent must be made using physical twin pointers.

Use of the LABEL field

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

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

You can use this capability in much the same manner as the CSECT statement of assembler language, with the following restrictions:

- A label used in the label field of a DATASET statement containing operands cannot be used on another DATASET statement containing operands.
- Labels must be alphanumeric and must be valid labels for an assembler language statement.
- Unlabeled DATASET statements must have operands.

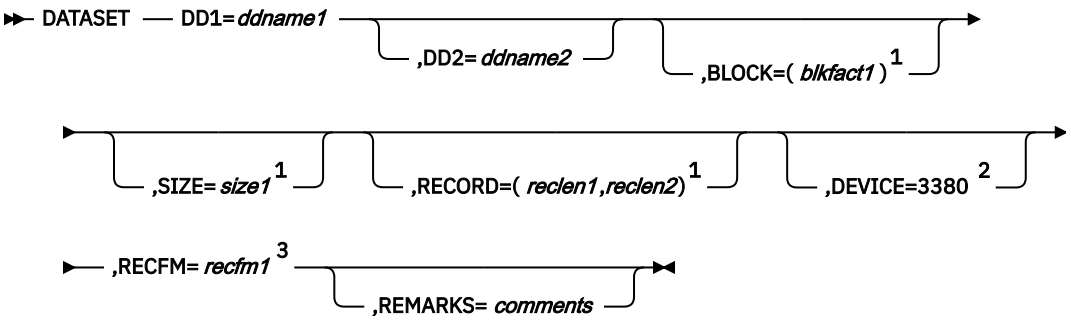
Table 3. Using the label field to group segment types

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

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

The format of the DATASET statement for each database type is shown in the following examples.

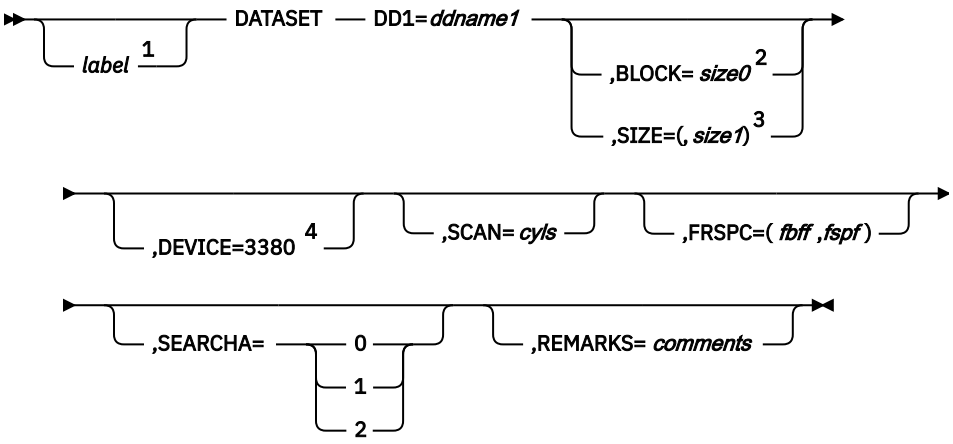
GSAM database DATASET Statement



Notes:

[1] If you do not specify a value, the DBDGEN utility generates the value used.
[2] The DEVICE parameter is ignored.
[3] RECFM is only valid for a GSAM database.

HDAM database DATASET statement



Notes:

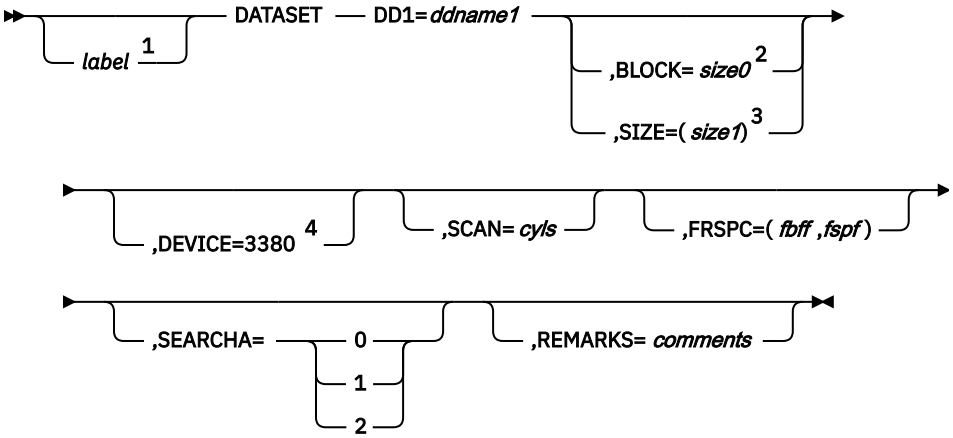
[1] If you have multiple DATASET statements with the same *label*, only the first one with the common *label* can contain operands that define the physical characteristics of the data set group.

[2] If you do not specify a value, the DBDGEN utility generates the value used.

[3] For VSAM, the valid parameter specifications for a SIZE keyword are 512 bytes, 1 KB, 2 KB, 4 KB, 8 KB, and multiples of 2 KB up to 28 KB.

[4] The DEVICE parameter is ignored.

HIDAM database DATASET statement



Notes:

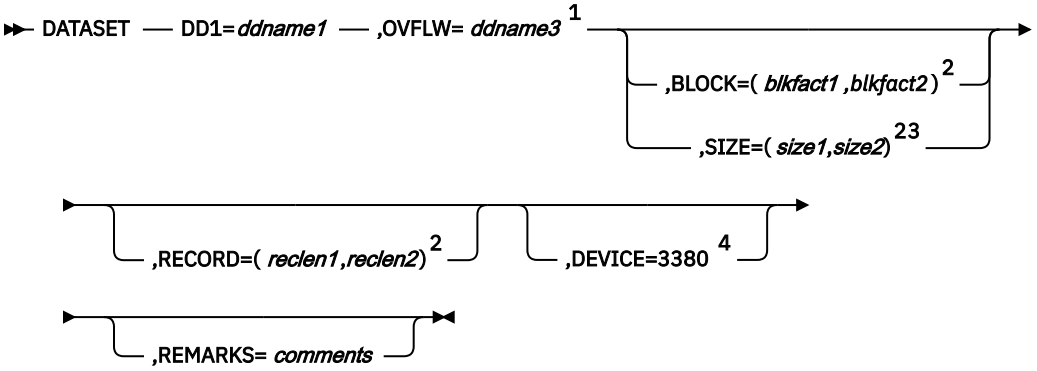
[1] If you have multiple DATASET statements with the same *label*, only the first one with the common *label* can contain operands that define the physical characteristics of the data set group.

[2] If you do not specify a value, the DBDGEN utility generates the value used.

[3] For VSAM, the valid parameter specifications for a SIZE keyword are 512 bytes, 1 KB, 2 KB, 4 KB, 8 KB, and multiples of 2 KB up to 28 KB.

[4] The DEVICE parameter is ignored.

HISAM/SHISAM database DATASET statement



Notes:

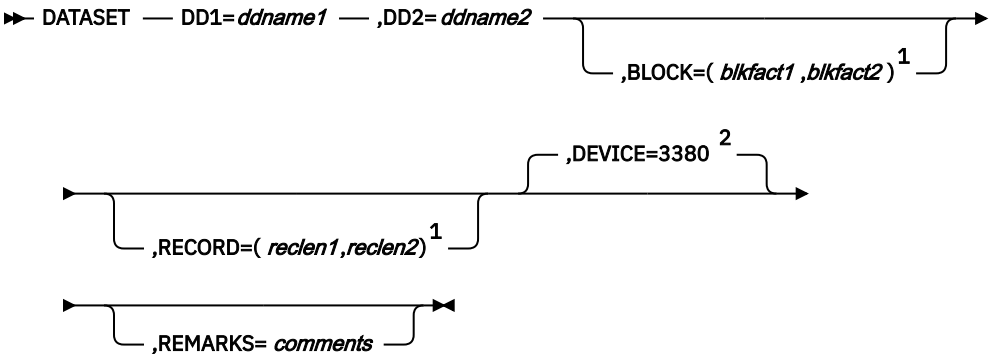
[1] If a HISAM database has only one segment type defined, you do not need to specify OVFLW. OVFLW is invalid in a simple HISAM database.

[2] If you do not specify a value, the DBDGEN utility generates the value used.

[3] The valid parameter specifications for a SIZE keyword are 512 bytes, 1 KB, 2 KB, 4 KB, 8 KB, and multiples of 2 KB up to 28 KB.

[4] The DEVICE parameter is ignored.

HSAM/SHSAM database DATASET statement

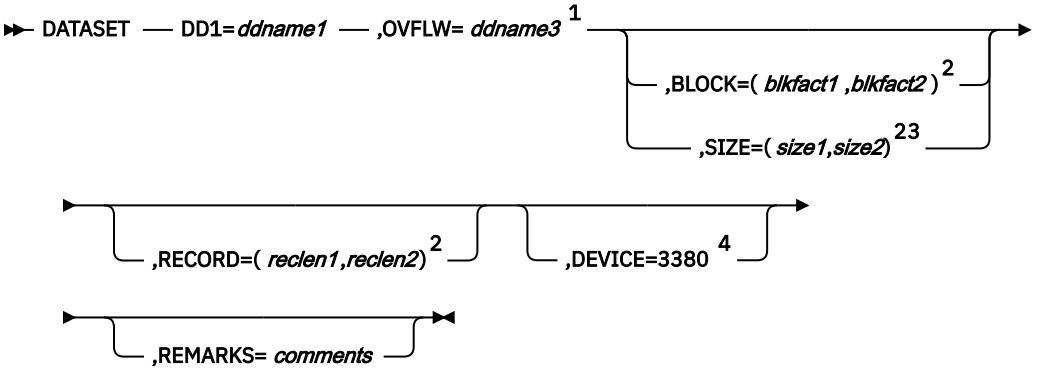


Notes:

[1] If you do not specify a value, the DBDGEN utility generates the value used.

[2] DEVICE parameter is ignored.

INDEX database DATASET statement

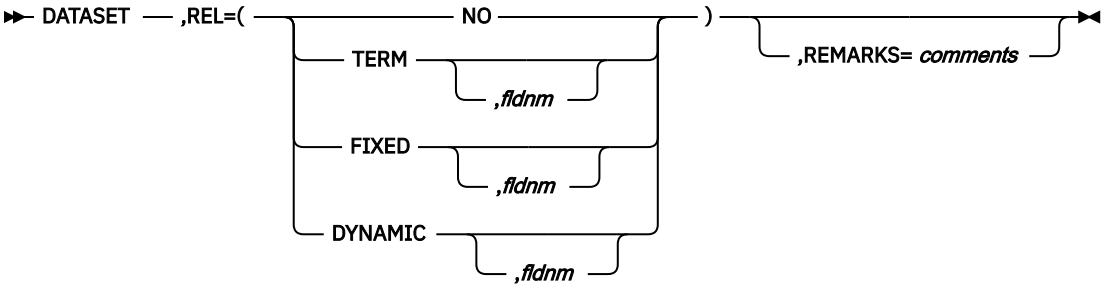


Notes:

[1] If the keys of all the index segments are unique, you do not need to specify OVFLW.
[2] If you do not specify a value, the DBDGEN utility generates the value used.
[3] The valid parameter specifications for a SIZE keyword are 512 bytes, 1 KB, 2 KB, 4 KB, 8 KB, and multiples of 2 KB up to 28 KB.
[4] The DEVICE parameter is ignored.

LOGICAL database DATASET statement

MSDB database DATASET statement



DATASET statement parameter description

DATASET
Identifies this as a DATASET control statement for a DL/I database.

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

DD1=
Specifies the ddname of the primary data set in this data set group. ddname1 must be a 1- to 8-character alphanumeric name. IMS use of the data set indicated by this parameter depends on the type of database being defined as shown in the following list:

Database Type
Use of the DD1= parameter

HSAM/SHSAM
ddname of input data set

GSAM
ddname of input data set

HISAM/SHISAM
ddname of primary data set in data set group

HIDAM
ddname of data set in data set group

HDAM
ddname of data set in data set group

MSDB
Parameter is invalid

DEDB
Name of defined area

INDEX
ddname of primary data set

LOGICAL
Parameter is invalid

For an HSAM, SHSAM, or GSAM database, this input data set is used when an application program retrieves data from the database.

DEVICE=
Specifies the physical storage device type on which the data set in this area is stored. The default is 3380. If you code any other device, it will be ignored.

DD2=
Specifies the 1- to 8-character alphanumeric ddname of the output data set required for an HSAM or SHSAM database and optional for a GSAM database. If it is omitted, ddname1 is assumed. This output data set is used by IMS when loading the database.

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

OVFLW=
Specifies the 1- to 8-character alphanumeric ddname of the overflow data set in this data set group. This parameter must be specified for:

- An INDEX database that contains index pointer segments with nonunique keys
- All data set groups of a HISAM database except when only one segment type is defined in the HISAM database

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

The following restrictions apply:

- The OVFLW parameter is not allowed when a simple HISAM database is defined.
- When a HISAM database that contains only one segment type is defined, the OVFLW parameter does not have to be specified.
- No OVFLW parameter on the DATASET statement is required for the index DBD because all index segments are inserted in the key sequenced data set of the index.
- If ACCESS=(INDEX,SHISAM) is specified, then the OVFLW parameter is invalid.

BLOCK=
Is used to specify the blocking factors (blkfact1, blkfact2) to be used for data sets in a data set group for HSAM, SHSAM, GSAM, HISAM, SHISAM, and INDEX databases, or is used to specify the block size or control interval size without overhead (size0) for the data set in a data set group for HDAM and HIDAM databases.

For HISAM, SHISAM, and INDEX databases that use VSAM as the access method, use the SIZE= parameter to specify control interval size in place of the BLOCK= parameter. If the SIZE= keyword is used for a HISAM, SHISAM, or INDEX database, the BLOCK= keyword is invalid.

The SIZE and BLOCK keywords are mutually exclusive for the following database access types:

- HDAM
- HIDAM
- HISAM
- INDEX
- SHISAM

In cases where the RECORD= and BLOCK= operands are used, the resulting control interval size must be a multiple of 512 when the resulting size is less than 8192 bytes. If the product of the record length specified times the blocking factor specified plus VSAM overhead is not a multiple of 512 and is less than 8192 bytes, the resulting control interval size is obtained by rounding the value up to the next higher multiple of 512. Control interval sizes from 8192 to 30720 bytes (maximum allowed size) must be in multiples of 2048 bytes. When the product of the RECORD= and BLOCK= operands plus VSAM overhead is from 8192 to 30720 bytes but is not a multiple of 2048, the resulting control interval size is obtained by rounding the value up to the next higher multiple of 2048.

The VSAM overhead is 7 bytes if the blocking factor is 1; otherwise, it is 10 bytes. The maximum block size for OSAM data sets is 32 KB.

For HDAM and HIDAM databases, the BLOCK= parameter is used to enable you to override DBDGEN's computation of control interval or block size. However, in addition to the value specified in the BLOCK= parameter, DBDGEN adds space for root anchor points, a free space anchor point, and access method overhead. The block or control interval size that results can be determined by referring to the equations in the description of the SIZE= parameter or by examining the output of DBDGEN. If SIZE= is not specified and the access method is VSAM, DBDGEN calculates the best VSAM LRECL value by equally distributing any unused space in the CI to each logical record in the CI. If SIZE= is specified or the database is SHISAM, this is not done.

The following table explains the use of the BLOCK= and RECORD= operands.

Table 4. BLOCK and RECORD operands

Database type	Use of BLOCK and RECORD operands
HSAM/SHSAM	**BLOCK=** blkfact1 applies to input data set and should always be 1. blkfact2 applies to output data set and should always be 1. **RECORD=** reclen1 is the input record length. reclen2 is the output record length. HSAM/SHSAM is always unblocked; LRECL and BLKSIZE are equal.
GSAM	**BLOCK=** blkfact1 applies to input/output data set. blkfact2 is an invalid subparameter. **RECORD=** reclen1 is the size of an LRECL length or maximum size for a variable length record. reclen2 is the minimum size for a variable length record. **SIZE=** size1 is the BLKSIZE for input/output data set. size2 is an invalid subparameter.

Table 4. BLOCK and RECORD operands (continued)

Database type	Use of BLOCK and RECORD operands
HISAM/SHISAM	**BLOCK=** blkfact1 is the primary data set blocking factor. blkfact2 is the overflow data set blocking factor. **RECORD=** reclen 1 is the data set logical record length. reclen2 is the overflow data set logical record length.
HIDAM, HDAM	**BLOCK=** size0 is size without overhead of OSAM or VSAM data set group **RECORD=** Is ignored.
MSDB	BLOCK= and RECORD= operands are invalid
DEDB	BLOCK= and RECORD= operands are invalid.
INDEX	**BLOCK=** blkfact1 is the primary data set blocking factor. blkfact2 is the overflow data set blocking factor. **RECORD=** reclen1 is the primary data set logical record length. reclen2 is the overflow data set logical record length.
LOGICAL	BLOCK= and RECORD= operands are invalid.

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

SIZE=

Overrides how the DBDGEN utility computes control interval or block size. If the value specified for SIZE= is different from the control interval size defined to VSAM using the Access Method Services, DL/I uses the value defined to VSAM.

For DL/I DBDs, you can effectively modify the DBD without a DBDGEN by redefining the control interval size to VSAM using the Access Method Services. This allows you to migrate databases to new devices without a DBDGEN. When used, no overhead is added to the values specified and the value specified is not validated by IMS.

For VSAM data sets, when the values specified are less than 8192, they must be a multiple of 512. If not a multiple of 512, DBDGEN rounds the value specified to the next higher multiple of 512 and issue a warning message. Values specified in the range of 8192 to 30720 bytes (maximum allowed size) must be a multiple of 2048. If not a multiple of 2048, DBDGEN rounds the value specified to the next higher multiple of 2048 and issue a warning message.

For HISAM, SHISAM, primary HIDAM index, and secondary index databases, size1 specifies the control interval or block size of the primary data set in a data set group, and size2 specifies the control interval or block size of the overflow data set.

For HDAM and HIDAM databases, only the size1 parameter is used. The size1 parameter specifies the control interval or block size of the data set in the data set group.

When SIZE is specified for a HISAM or INDEX database, the RECORD parameter must also be specified; the size value specified must be a multiple of the record parameter in order to allow

VSAM to open the data sets involved. Following are equations that show the minimum block or control interval size that you can specify for databases.

The maximum block size of OSAM data sets is 32768 bytes. An OSAM data set with an even length block size has an 8-gigabyte size limit. If the database is saved with image copy, 32752 bytes is the maximum amount that can be specified for the block size. Image copy processing module DFSUDMP0 adds 15 bytes to the block size for double-word alignment of its prefix, and the block size cannot exceed 32768. If the DBDGEN utility specifies the block size, 32752 bytes is the maximum amount specified.

Important: Calculating SIZE= for HISAM primary data set groups, primary HIDAM index data set groups, and secondary index data set groups

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

- primary size ≥ ROOTSEG + OVERHEAD + VSAM CONTROL
- overflow size ≥ MAXSEG + OVERHEAD + VSAM CONTROL

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

OVERHEAD=
Number of bytes required is:

7
Used for OSAM, if the database has more then one physical segment type

3
Used for OSAM, if the database has only one physical segment type

4
Used for INDEX VSAM databases with nonunique root segment keys

0
Used for INDEX VSAM databases unique root segment keys, not created using DL/I DOS.

5 bytes for all other VSAM databases.

VSAM CONTROL=
Number of bytes required is:

0
Used for OSAM, if the blocking factor is 1

7
Used for VSAM if the blocking factor is 1

10
Used for all other cases

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

Calculating SIZE= for HDAM Primary Data Set Group

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

- If rbn is specified, then the following two conditions must be met:
  - size ≥ (RAPs*4) + FSEAP + 2 + VSAM CONTROL
  - size ≥ MAXSEG + FSEAP + VSAM CONTROL
- If rbn is not specified, then the following condition must be met:

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

RAPs=
Number of root anchor points specified for the root addressable area of the database.

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

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

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

Calculating SIZE= for HDAM Secondary Data Set Groups

The block or control interval size specified must be:

size ≥ MAXSEG + FSEAP + VSAM CONTROL

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

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

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

Calculating SIZE= for HIDAM Data Set Groups

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

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

size ≥ MAXSEG + FSEAP + RAP + VSAM CONTROL

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

size ≥ MAXSEG + FSEAP + VSAM CONTROL

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

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

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

RAP=
4 bytes for one root anchor point.

RECORD=(*reclen1,reclen2*)
Specifies the data management logical record lengths to be used for this data set group. This parameter is optional and cannot be specified if ACCESS=LOGICAL is used on the DBD statement. reclen1 and reclen2 must be numeric values. The value of reclen2 must always be equal to or greater than the value of reclen1 except for GSAM databases. The meaning of each of the parameter's parameters depends on the type of database being defined.

For a simple HISAM (SHISAM) database, the logical record length specified must be the same as the segment length specified. The minimum allowable logical record lengths for HISAM and INDEX DBDs are the same as the minimum block or control interval sizes described for the DATASET SIZE= parameter, except that VSAM CONTROL should be ignored. In addition, for both the VSAM KSDS and ESDS for HISAM, and INDEX DBDs, the logical record length specified must also be an even value.

For VSAM primary index (INDEX, VSAM) databases, the overflow logical record length (reclen2) parameter should not be defined, because all index segments are inserted into the key sequence data set.

For a GSAM database, reclen1 specifies the size of a logical record for a fixed-length record or the maximum size for a variable-length or undefined record. The value of reclen2 specifies the minimum size for a variable-length or undefined record. For variable length GSAM/BSAM database, IMS adds 2 bytes to the record length value in the GSAM records passed by the application. This is done in order to accommodate the ZZ field that makes up the BSAM Record Descriptor Word (RDW) when the record is written to the I/O device.

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

F
The records are fixed-length.

FB
The records are fixed-length and blocked.

V
The records are variable-length.

VB
The records are variable-length and blocked.

U
The records are of undefined length.

This parameter is only valid for a GSAM database.

SCAN=*cyls*
Specifies the number of direct-access device cylinders to be scanned when searching for available storage space during segment insertion operations. This parameter is optional. It is only used when this DBD generation defines a HIDAM or HDAM database. If specified, *cyls* must be a decimal integer whose value does not exceed 255. Typical values are from 0 to 5. The default value is 3. If SCAN=0 is specified, only the current cylinder is scanned for space.

Scanning is performed in both directions from the current cylinder position. If a scan limit value causes scanning to include an area outside of the current extent, IMS adjusts the scan limits so that scanning does not exceed current extent boundaries. If space cannot be found for segment insertion within the cylinder bounds defined by this parameter, space is used at the current end of the data set group for the database.

FRSPC=
Specifies how free space is to be distributed in an HDAM or HIDAM database. 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 database load or reorganization (where fbff=n). The range of fbff includes all integer values from 0 to 100 except fbff=1. The fspf is the free space percentage factor. It specifies the minimum percentage of each control interval or block that is to be left as free space in this data set group. The range of fspf is from 0 to 99. The default value for fbff and fspf is 0. If the total of the percentage of free space specified and any segment size exceeds the control interval or block size, a warning message that flags oversized segments is issued by DBDGEN. When loading oversized segments, the "fspf" specification is ignored and one control interval or block is used to load each oversized segment.

When you specify the first parameter, FBFF, realize that a smaller value increases the frequency of free space in the database. A value of 2, for example, would mean that after every piece of data there would be a free space block. This causes system performance degradation when running reorganization or load utilities because of the extra processing required for the free space blocks.

SEARCHA=
Specifies the type of HD space search algorithm that IMS uses to insert a segment into an HD database.

0
Specifies that IMS chooses which HD space search algorithm to use. This is the default. For this release, IMS uses the same algorithm it would use if you had specified SEARCHA=2.

1
Specifies that IMS uses the HD space search algorithm that does not search for space in the second-most desirable block or CI.

2
Specifies that IMS uses the HD space search algorithm that includes a search for space in the second-most desirable block or CI.

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

MSDBs with terminal-related keys are not supported for ETO in IMS V5 or above. Other types of MSDBs are still supported.

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

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

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

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

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

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

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

REMARKS=
Optional user comments. A 1- to 256-character field.

If your comments contain special characters, such as commas or blank spaces, enclose the full comment string in single quotation marks.

The value specified on the REMARKS keyword cannot contain the following characters:

- Single quotation marks, except when they are used to enclose the full comment string. If a single quotation mark is entered before the end of the full comment string, the remainder of the comment string is truncated. The following examples show correct and incorrect usages of single quotation marks on the REMARKS keyword:

  CORRECT
  ```
  REMARKS='These remarks apply to the XYZ application'
  ```

  INCORRECT
  ```
  REMARKS='These remarks apply to the 'XYZ' application'
  ```

- Double quotation marks.
- Less than (<) symbols.
- Greater than (>) symbols.
- Ampersands (&).

Data sets in IMS data set groups

The DD statements for non-HALDB data sets in each IMS database must be provided with each job that accesses the database. For databases used by message or batch message processing programs, you must include DD statements in the JCL for the IMS control region. For databases used exclusively in the batch processing environment, you must include DD statements in the JCL for the batch processing region. In a z/OS online environment, databases can be dynamically allocated.

DD statements are not required for HALDB data sets, because they are dynamically allocated.

DD statements required for VSAM

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

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

UNIT=, VOLSER=, and SPACE= parameters are not required because all VSAM data sets are cataloged.

For a HISAM database, two DD statements are required: one for the KSDS and one for the ESDS. If the HISAM database has only one segment type defined, only the KSDS DD statement is required.

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

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

For secondary index databases with nonunique keys, two DD statements are required; one for the KSDS and one for the ESDS. Note that secondary index databases with nonunique keys are not supported for HALDB. In addition to the DD statements defining VSAM data sets, a DD statement specifying a data set containing parameters defining the IMS VSAM buffer pool must be provided for batch regions. The DDNAME for this DD statement is DFSVSAMP. For online IMS execution, this information is provided in a member of the IMS.PROCLIB data set with member name DFSVSMxx.

DD statements required for OSAM

For HSAM or SHSAM, you must provide a DD statement for either input or output in the following format:

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

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

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

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

For HDAM or HIDAM, a DD statement is required for the OSAM data set of each data set group. The format is as follows:

```
//dd1      DD  DSNAME=         ,UNIT=      ,VOL=SER=   ,
//             DISP=       ,DCB=(DSORG=PS[,OPTCD=W])
```

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

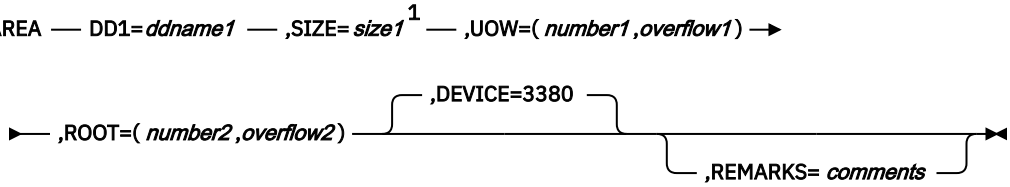
If a model DSCB is to be used to describe a generation data set, the LRECL, RECFM, and BLKSIZE parameters must be omitted from the model DSCB. This information is obtained from the DBD and cannot be overridden.

AREA statement

DEDB databases use an AREA statement to define an area within a database.

In the DBDGEN input deck for a DEDB, all AREA statements must be placed between the DBD statement and the first SEGM statement. At least one AREA statement is required, but as many as 2048 AREA statements can be used to define multiple areas.

Restriction: AREA statements are not allowed for HALDB databases. Partitions are defined outside DBDGEN.



Notes:

[1] The valid parameter specifications for a DEDB SIZE keyword are 512 bytes, 1 KB, 2 KB, 4 KB, 8 KB, and multiples of 4 KB up to 28 KB. To ensure future compatibility, use only CI sizes that are multiples of 4 KB.

AREA statement parameter description

AREA
Identifies this statement as a DEDB AREA control statement.

DD1=
Specifies the ddname of the defined area. ddname1 must be a 1- to 8-character alphanumeric (A-Z, 0-9, #, @, $) name. This parameter can be an area name or a ddname for single area data sets but can only be an area name for multiple area data sets. If the database is registered in DBRC, this parameter should specify the area name.

DEVICE=
Specifies the physical storage device type on which the data set in this area is stored. The default is 3380. If you code any other device, it will be ignored.

SIZE=
Specifies the control interval. Size can be 512 bytes, 1 KB, 2 KB, 4 KB, and 8 KB and multiples of 4 KB up to 28 KB. For future compatibility, only CI sizes that are multiples of 4 KB should be used. No default value is allowed.

Restriction: 4 KB cannot be specified with a 2319 device.

For DEDBs, the DBDGEN SIZE= must match the control interval size defined to VSAM, because IMS uses this value in accessing the data set. If the control interval size is changed in the VSAM data set, the DBD for that area must be changed to the new SIZE= value.

UOW=
Specifies the number of control intervals in a unit of work (UOW). The UOW= parameter has two operands, number1 and overflow1.

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

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

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

REMARKS=
Optional user comments. A 1- to 256-character field.

If your comments contain special characters, such as commas or blank spaces, enclose the full comment string in single quotation marks.

The value specified on the REMARKS keyword cannot contain the following characters:

- Single quotation marks, except when they are used to enclose the full comment string. If a single quotation mark is entered before the end of the full comment string, the remainder of the comment string is truncated. The following examples show correct and incorrect usages of single quotation marks on the REMARKS keyword:

  CORRECT
  ```
  REMARKS='These remarks apply to the XYZ application'
  ```

  INCORRECT
  ```
  REMARKS='These remarks apply to the 'XYZ' application'
  ```
- Double quotation marks.
- Less than (<) symbols.
- Greater than (>) symbols.
- Ampersands (&).

ROOT=
Specifies characteristics of a DEDB area. The ROOT= parameter has two operands, number2 and overflow2.

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

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

The reorganization UOW is automatically allocated by the DEDB Initialization utility. VSAM space definition should include this additional UOW. That is, the total space required is the root addressable area, the independent overflow, and one additional UOW for reorganization. The reorganization UOW is not used by the High-Speed DEDB Direct Reorganization utility, but may be used by other functions of IMS.

Example: This example allocates 2048*64*936 bytes and leaves the rest of the area for sequential dependent segments.

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

Because there is only one root anchor point (RAP) per control interval, the total number of RAPs within the area is given by: (64-14)*(936-36) = 45000 RAPs.

The amount of space allocated for independent overflow by DBDGEN can be increased while IMS is online.

SEGM statements

The SEGM statement defines a segment type, the segment's position in a database hierarchy, the physical characteristics of the segment, and how the segment is to be related to other segments.

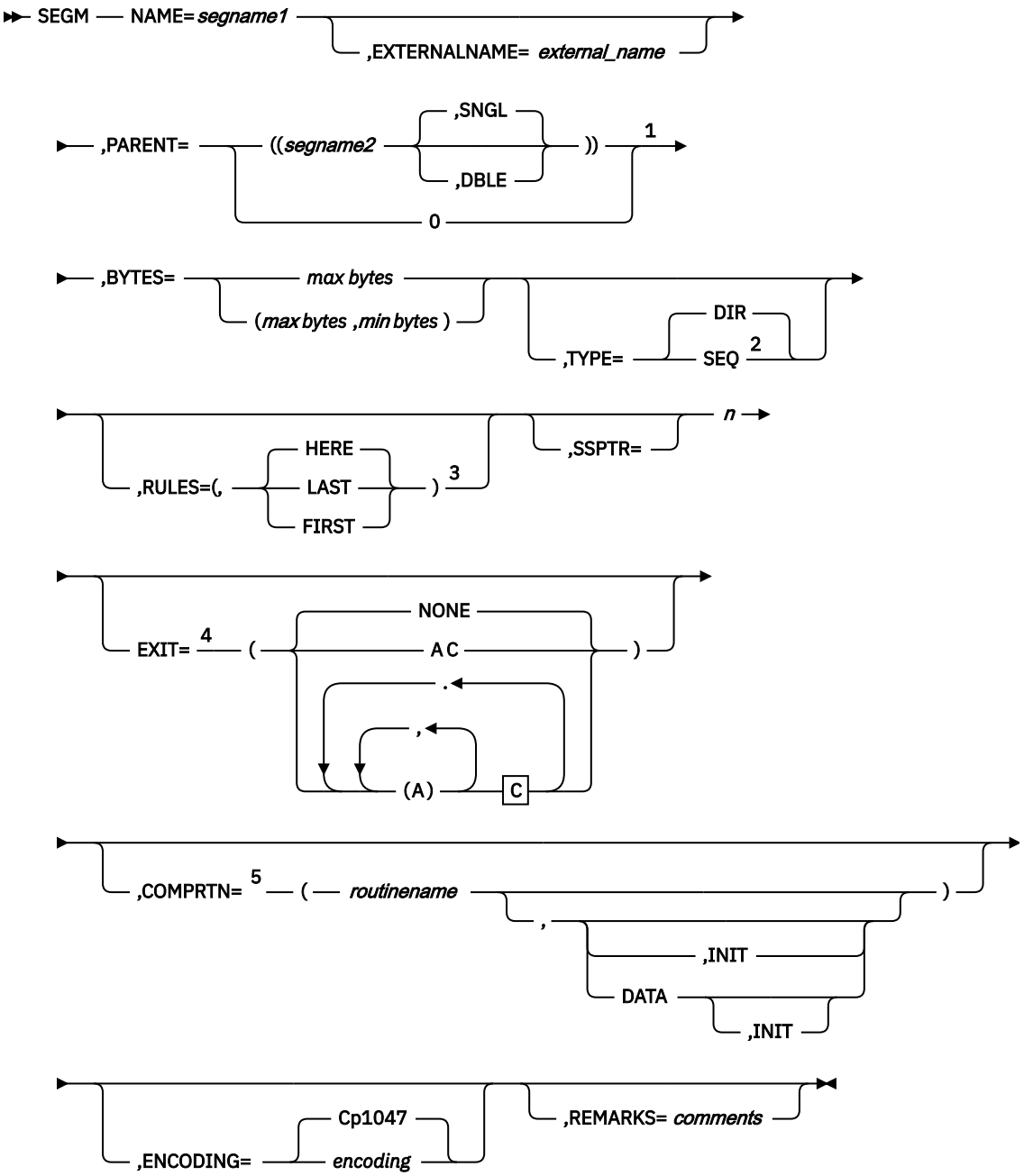
Except for GSAM databases, at least one SEGM statement must immediately follow each DATASET statement; the segment that is defined by the SEGM statement is placed in the data set group that is defined by the DATASET statement. Except for MSDBs and DEDBs, a maximum of 255 SEGM statements are allowed in a DBD generation. For an MSDB, only one SEGM statement can be specified. For a DEDB, at least one and up to 127 SEGM statements must immediately follow the last AREA statement; no other SEGM statements can be provided in the DBD generation. SEGM statements must be placed in the input file in hierarchic sequence, and a maximum of 15 hierarchic levels can be defined.

For a GSAM database, you can specify one SEGM statement in the DBD generation. The SEGM statement defines a virtual segment that you can use to define non-searchable fields that would otherwise be mapped only in a COBOL copybook. The resulting field metadata is stored in the IMS catalog for use by application programs and products that access the GSAM database through the IMS Universal drivers.

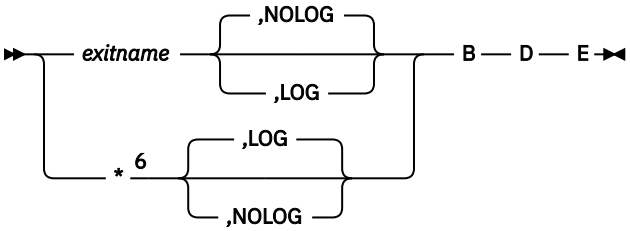
The SEGM statement is used with FIELD, XDFLD, and LCHILD statements to totally define a segment to IMS. The FIELD statement defines fields within segments, the XDFLD statement defines fields used for secondary indexing, and the LCHILD statement defines index or logical relationships between segments.

The format of the SEGM statement for each database type is shown in the following examples.

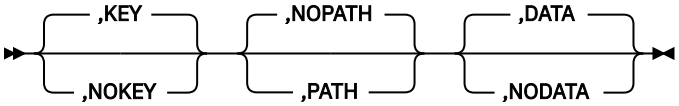
DEDB database SEGM statement


SEGM
NAME=segname1
,EXTERNALNAME= external_name
,PARENT=
((segname2
,SNGL
,DBLE
))
1
0
,BYTES=
max bytes
(max bytes ,min bytes)
,TYPE=
DIR
SEQ
2
,RULES=(,
HERE
LAST
FIRST
)
3
,SSPTR=
n
EXIT=
4
(
NONE
A C
.
,
(A)
C
)
,COMPRTN=
5
(
routinename
,
,INIT
DATA
,INIT
)
,ENCODING=
Cp1047
encoding
,REMARKS= comments

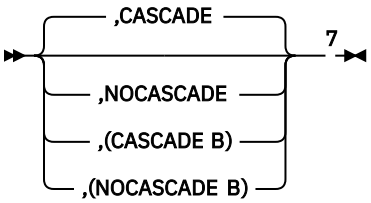
A


exitname
,NOLOG
,LOG
B
D
E
*
6
,LOG
,NOLOG

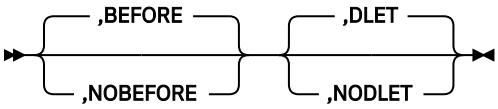
B


,KEY
,NOKEY
,NOPATH
,PATH
,DATA
,NODATA

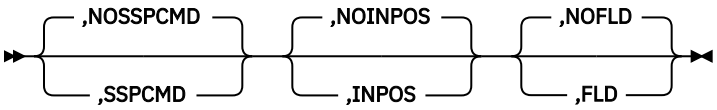
C



D



E



Notes:

[1] The PARENT= keyword can be omitted, or PARENT=0 can be specified for the root segment type of a database.

[2] TYPE=SEQ is required on SEGM statements for the sequential dependent type.

[3] Required when a segment type does not have a unique sequence field. HERE is the default. When using Fast Path sequential dependent segment processing, the insert rule of FIRST is always used and cannot be overridden. For DEDB direct dependent segment processing, HERE is the default.

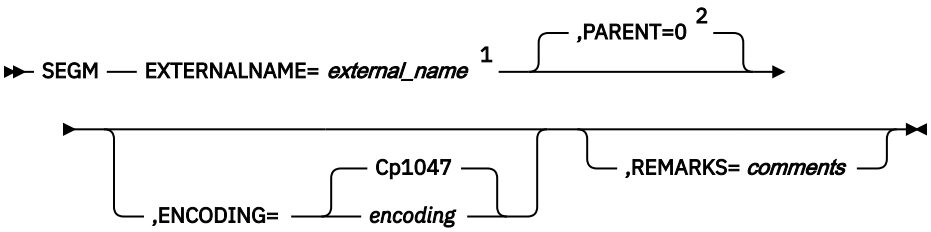
[4] Used for the Data Capture exit routine. You can specify more than one exit routine on a SEGM statement.

[5] Used for Segment Edit/Compression exit routine.

[6] If an exit routine name is not required because only logging is requested, specify the exit name as an asterisk (*). The default logging parameter in this case is LOG. If you specify an exit routine name, the default logging parameter is NOLOG.

[7] Used to control the CASCADE options. If any (CASCADE B) suboptions are specified with NOCASCADE in an **EXIT=** parameter, they will be ignored, and the value for those suboptions will be set to NO*xxxx*.

GSAM database SEGM statement

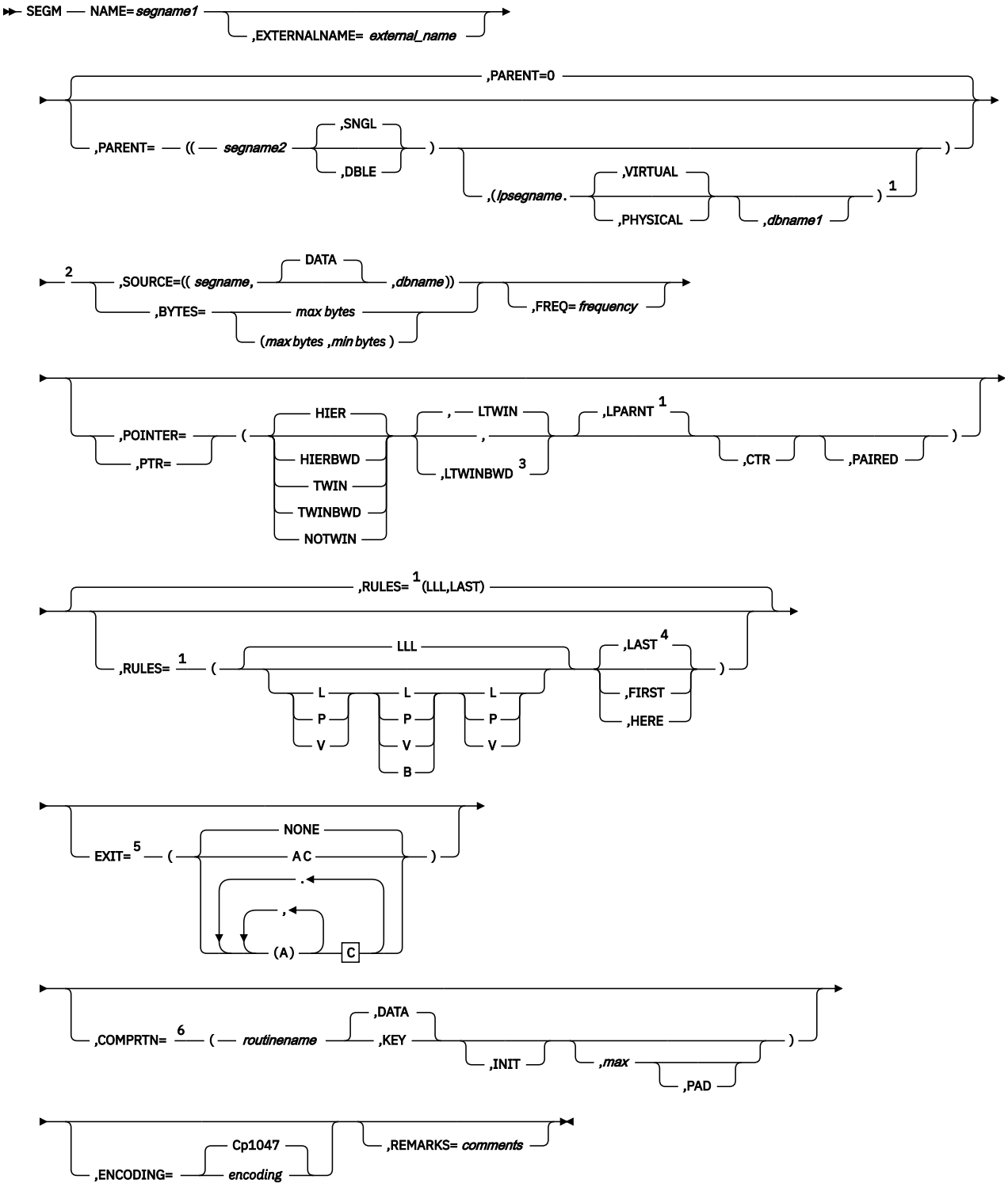


Notes:

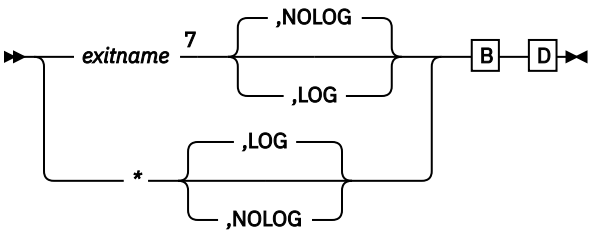
[1] NAME= is invalid for GSAM. EXTERNALNAME is required.

[2] The PARENT= keyword can be omitted, or PARENT=0 can be specified for the root segment type of a database.

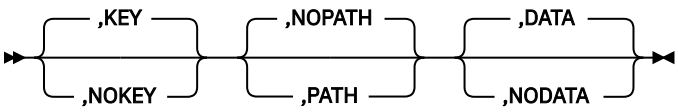
HDAM database SEGM statement


SEGM
NAME=segname1
,EXTERNALNAME= external_name
,PARENT=0
,PARENT=
((
segname2
,SNGL
,DBLE
)
,(lpsegname.
,VIRTUAL
,PHYSICAL
,dbname1
)
1
)
2
,SOURCE=((segname,
DATA
,dbname))
,BYTES=
max bytes
(max bytes ,min bytes)
,FREQ= frequency
,POINTER=
,PTR=
(
HIER
HIERBWD
TWIN
TWINBWD
NOTWIN
, LTWIN
,
,LTWINBWD
3
,LPARNT
1
,CTR
,PAIRED
)
,RULES=
1
(LLL,LAST)
,RULES=
1
(
LLL
L
P
V
L
P
V
L
P
V
B
,LAST
4
,FIRST
,HERE
)
EXIT=
5
(
NONE
AC
.
,
(A)
C
)
,COMPRTN=
6
(
routinename
,DATA
,KEY
,INIT
,max
,PAD
)
,ENCODING=
Cp1047
encoding
,REMARKS= comments

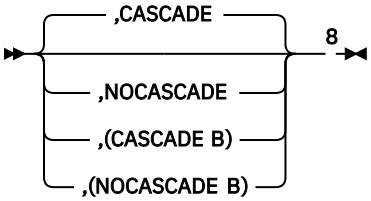
A


exitname
7
,NOLOG
,LOG
B
D
*
,LOG
,NOLOG

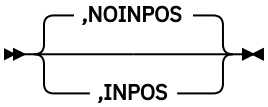
B



C



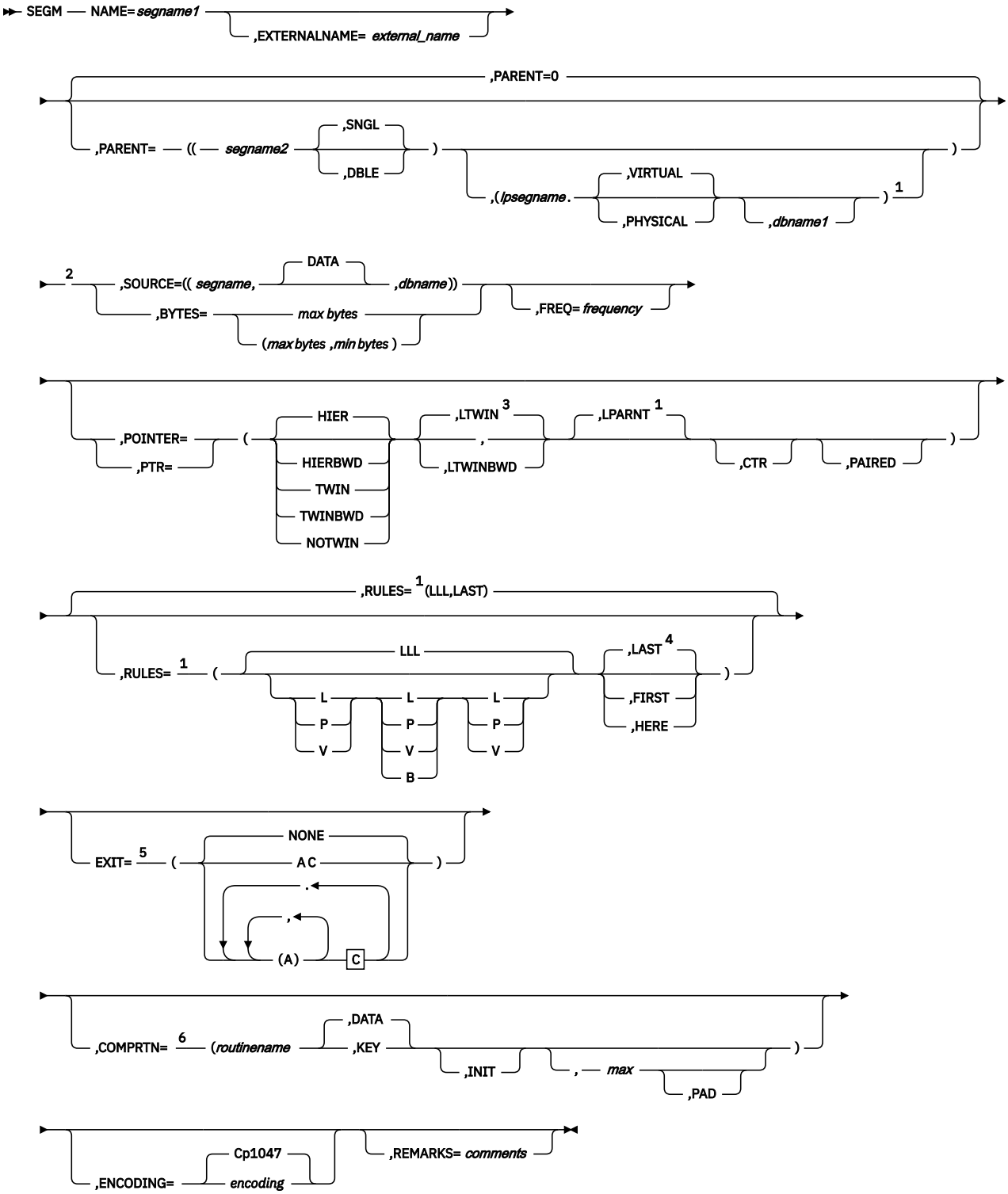
D



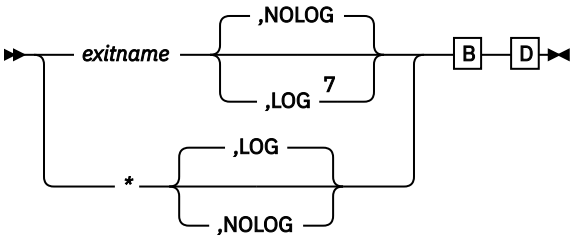
Notes:

[1] Optional for HDAM logical relationships.
[2] The PARENT= keyword can be omitted, or PARENT=0 can be specified for the root segment type of a database.
[3] Required for HDAM logical relationships; otherwise, it is optional.
[4] Required when a segment type does not have a unique sequence field. LAST is the default.
[5] Used for the Data Capture exit routine. You can specify more than one exit routine on a SEGM statement.
[6] Used for Segment Edit/Compression exit routine.
[7] If an exit routine name is not required because only logging is requested, specify the exit name as an asterisk (*). The default logging parameter in this case is LOG.
[8] Used to control the CASCADE options. If any (CASCADE B) suboptions are specified with NOCASCADE in an **EXIT=** parameter, they will be ignored, and the value for those suboptions will be set to NO*xxxx*.

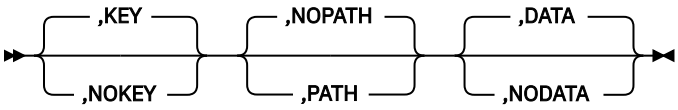
HIDAM database SEGM statement


SEGM
NAME=segname1
,EXTERNALNAME= external_name
,PARENT=0
,PARENT=
((
segname2
,SNGL
,DBLE
)
,(lpsegname.
,VIRTUAL
,PHYSICAL
,dbname1
)1
)
2
,SOURCE=((segname,
DATA
,dbname))
,BYTES=
max bytes
(max bytes ,min bytes)
,FREQ= frequency
,POINTER=
,PTR=
(
HIER
HIERBWD
TWIN
TWINBWD
NOTWIN
,LTWIN 3
,
,LTWINBWD
,LPARNT 1
,CTR
,PAIRED
)
,RULES= 1(LLL,LAST)
,RULES= 1
(
LLL
L
P
V
L
P
V
B
L
P
V
,LAST 4
,FIRST
,HERE
)
EXIT= 5
(
NONE
AC
.
,
(A)
C
)
,COMPRTN= 6
(routinename
,DATA
,KEY
,INIT
, max
,PAD
)
,ENCODING=
Cp1047
encoding
,REMARKS= comments

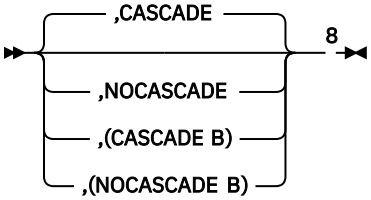
A


exitname
,NOLOG
,LOG 7
B
D
*
,LOG
,NOLOG

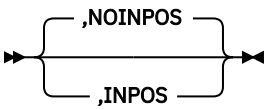
B



C



D



Notes:

[1] Optional for HIDAM logical relationships.

[2] The PARENT= keyword can be omitted, or PARENT=0 can be specified for the root segment type of a database.

[3] Required for HIDAM logical relationships; otherwise, it is optional.

[4] Required when a segment type does not have a unique sequence field. LAST is the default. When using Fast Path sequential dependent segment processing, the insert rule of FIRST is always used and cannot be overridden.

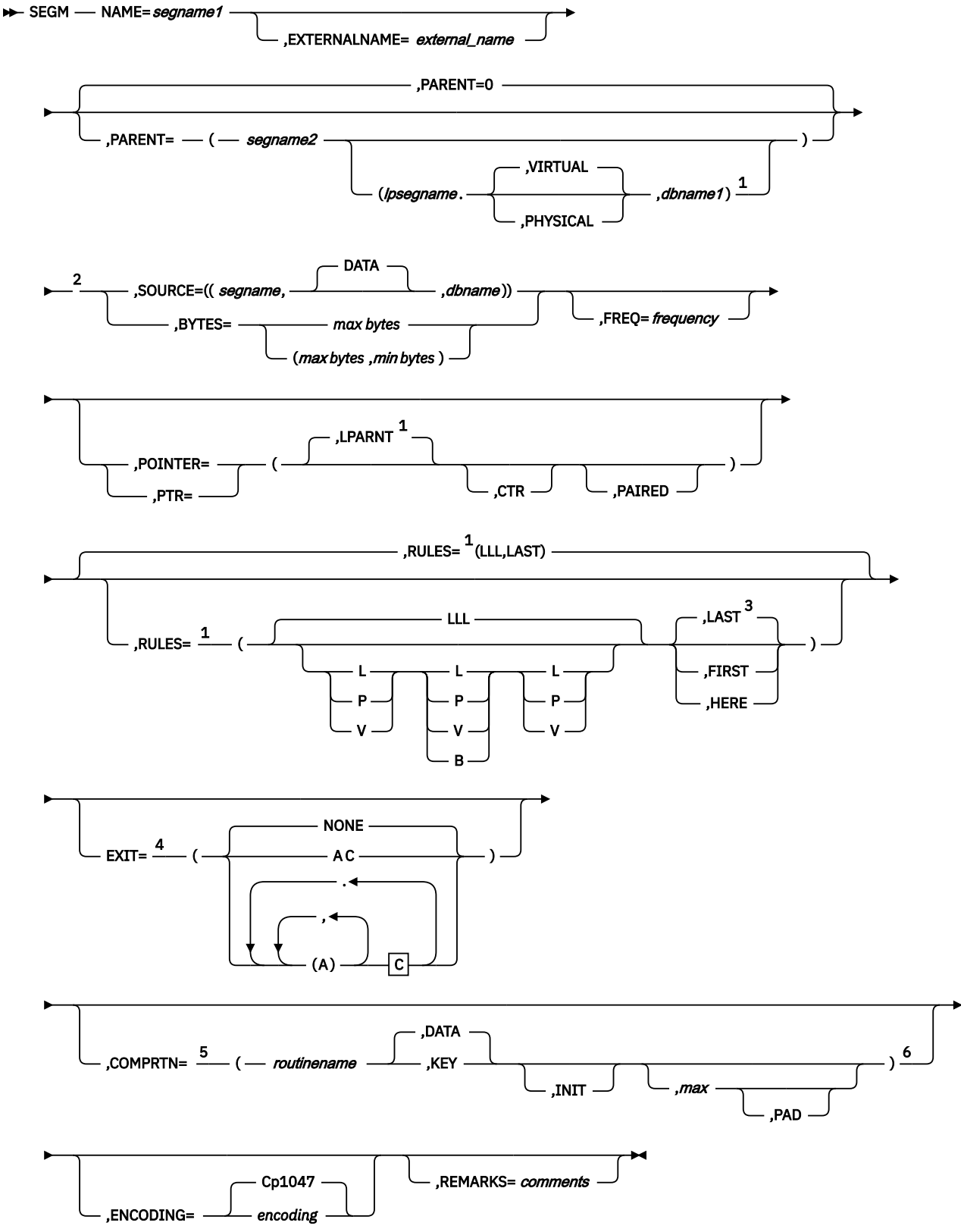
[5] Used for the Data Capture exit routine. You can specify more than one exit routine on a SEGM statement.

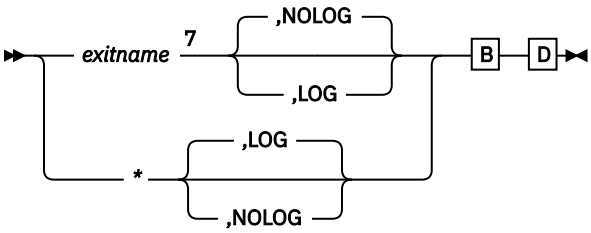
[6] Used for Segment Edit/Compression exit routine.

[7] If an exit routine name is not required because only logging is requested, specify the exit name as an asterisk (*). The default logging parameter in this case is LOG.

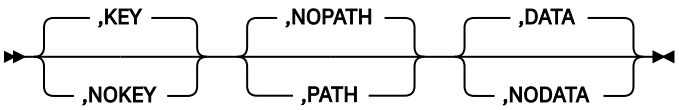
[8] Used to control the CASCADE options. If any (CASCADE B) suboptions are specified with NOCASCADE in an **EXIT=** parameter, they will be ignored, and the value for those suboptions will be set to NO*xxxx*.

HISAM/SHISAM database SEGM statement

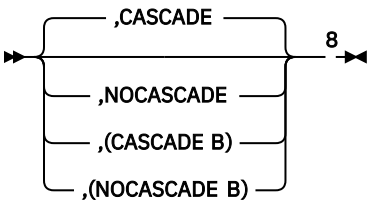

SEGM
NAME=segname1
,EXTERNALNAME= external_name
,PARENT=0
,PARENT=
(
segname2
(lpsegname.
,VIRTUAL
,PHYSICAL
,dbname1)
1
)
2
,SOURCE=((segname,
DATA
,dbname))
,BYTES=
max bytes
(max bytes ,min bytes)
,FREQ= frequency
,POINTER=
,PTR=
,LPARNT
1
,CTR
,PAIRED
,RULES= 1 (LLL,LAST)
,RULES=
1
LLL
L
P
V
,LAST
3
,FIRST
,HERE
EXIT=
4
NONE
AC
.
,
(A)
C
,COMPRTN=
5
routinename
,DATA
,KEY
,INIT
,max
,PAD
6
,ENCODING=
Cp1047
encoding
,REMARKS= comments
A



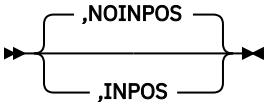
B



C



D



Notes:

[1] Required for HISAM logical relationships; otherwise, it is optional.

[2] The PARENT=keyword can be omitted, or PARENT=0 can be specified for the root segment type of a database.

[3] Required when a segment type does not have a unique sequence field. LAST is the default.

[4] Used for the Data Capture exit routine. You can specify more than one exit routine on a SEGM statement.

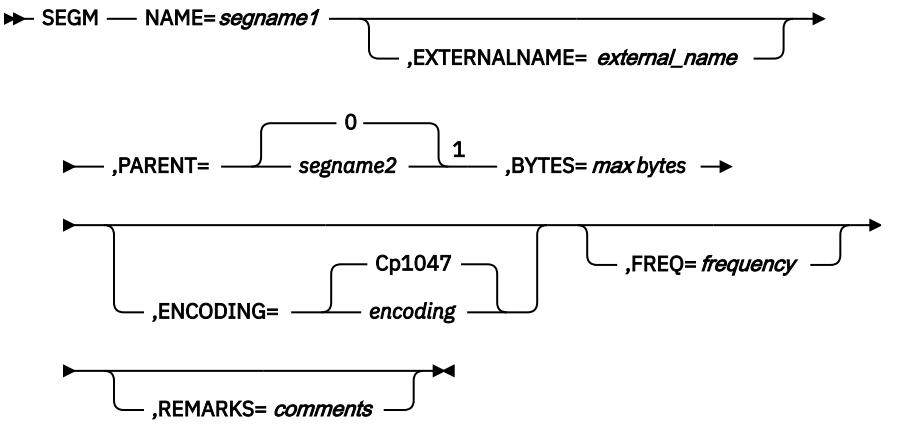
[5] Used for Segment Edit/Compression exit routine.

[6] Variable-length segments and segment edit/compression cannot be specified for a simple HISAM database.

[7] If an exit routine name is not required because only logging is requested, specify the exit name as an asterisk (*). The default logging parameter in this case is LOG.

[8] Used to control the CASCADE options. If any (CASCADE B) suboptions are specified with NOCASCADE in an **EXIT=** parameter, they will be ignored, and the value for those suboptions will be set to NO*xxxx*.

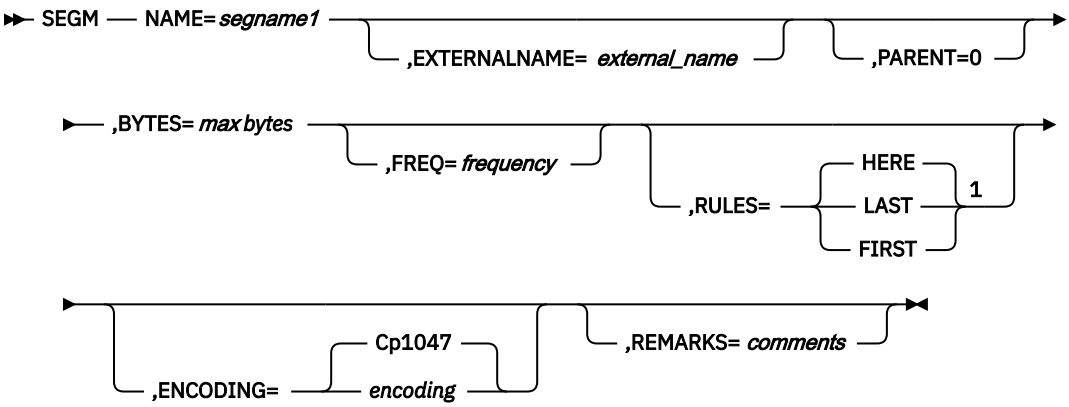
HSAM database SEGM statement



Notes:

[1] The PARENT= keyword can be omitted, or PARENT=0 can be specified for the root segment type of a database.

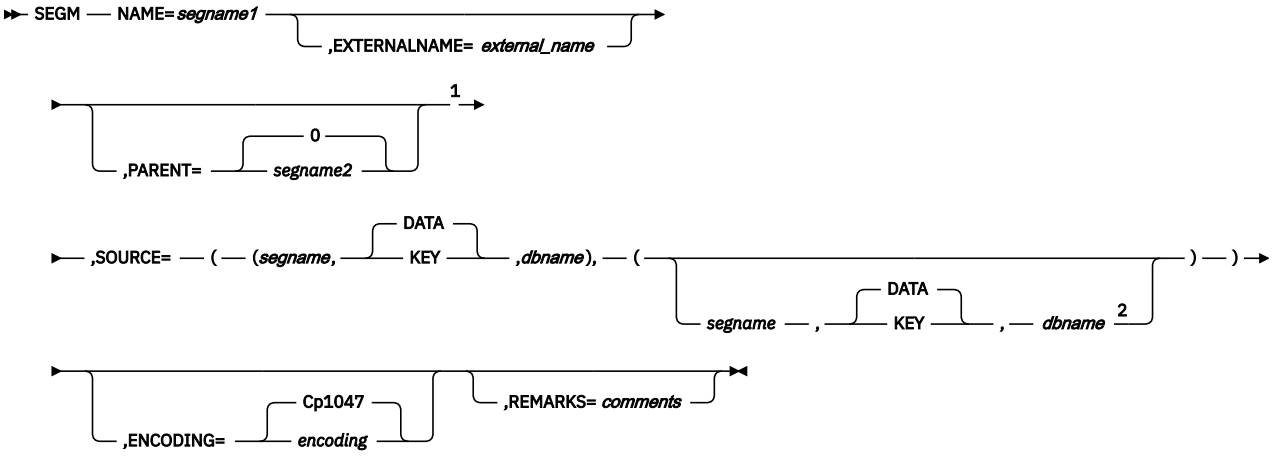
INDEX database SEGM statement



Notes:

[1] Required when a segment type does not have a unique sequence field. HERE is the default. When using Fast Path sequential dependent segment processing, the insert rule of FIRST is always used and cannot be overridden. For DEDB direct dependent segment processing, HERE is the default.

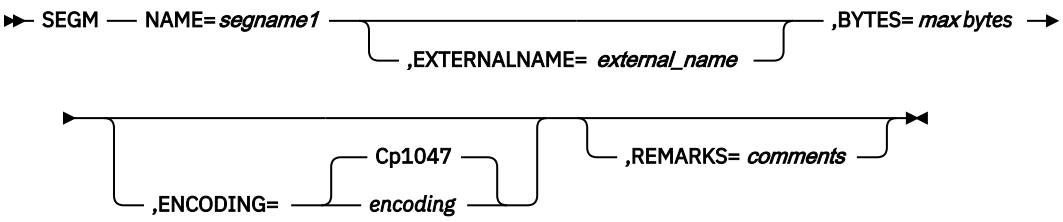
LOGICAL database SEGM statement



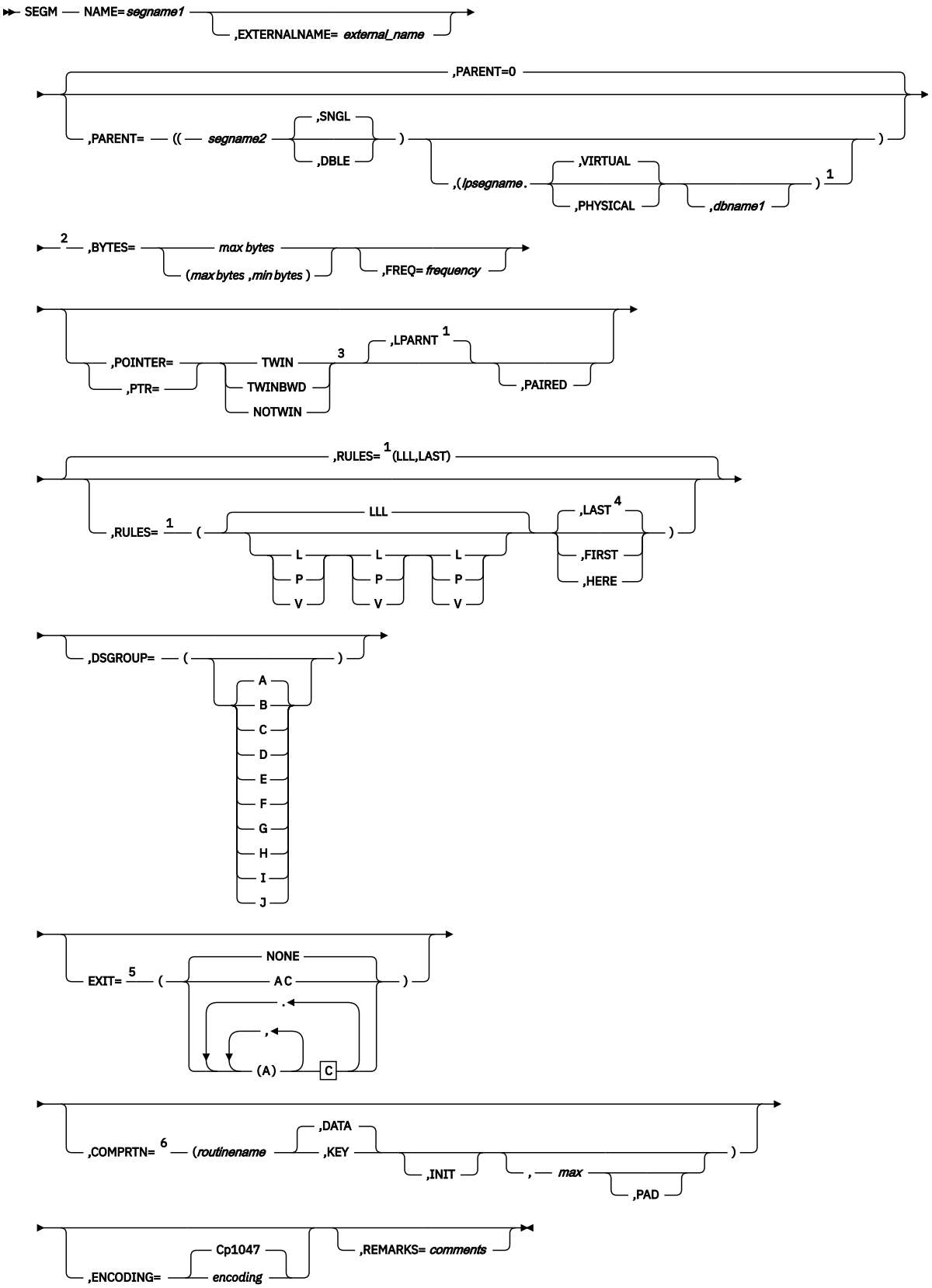
Notes:

[1] The PARENT= keyword can be omitted, or PARENT=0 can be specified for the root segment type of a database.
[2] Required when defining a concatenated segment type.

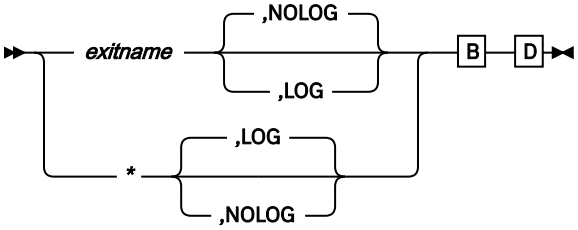
MSDB database SEGM statement



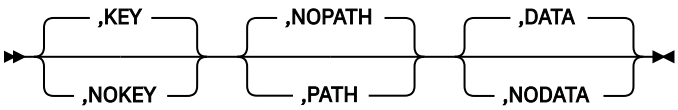
PHDAM database SEGM statement


SEGM
NAME=segname1
,EXTERNALNAME= external_name
,PARENT=0
,PARENT=
((
segname2
,SNGL
,DBLE
)
,(lpsegname.
,VIRTUAL
,PHYSICAL
,dbname1
)1
)
2
,BYTES=
max bytes
(max bytes ,min bytes)
,FREQ= frequency
,POINTER=
,PTR=
TWIN
TWINBWD
NOTWIN
3
,LPARNT 1
,PAIRED
,RULES= 1 (LLL,LAST)
,RULES= 1 (
LLL
L
P
V
,LAST 4
,FIRST
,HERE
,DSGROUP= (
)
A
B
C
D
E
F
G
H
I
J
EXIT= 5 (
NONE
AC
.
,
(A)
C
,COMPRTN= 6 (routinename
,DATA
,KEY
,INIT
, max
,PAD
,ENCODING=
Cp1047
encoding
,REMARKS= comments

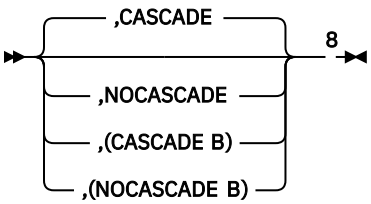
A



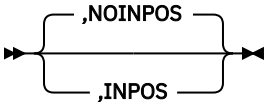
B



C



D



Notes:

[1] Optional for PHDAM logical relationships.

[2] The PARENT= keyword can be omitted, or PARENT=0 can be specified for the root segment type of a database.

[3] Required for PHDAM logical relationships; otherwise, it is optional.

[4] Required when a segment type does not have a unique sequence field. LAST is the default.

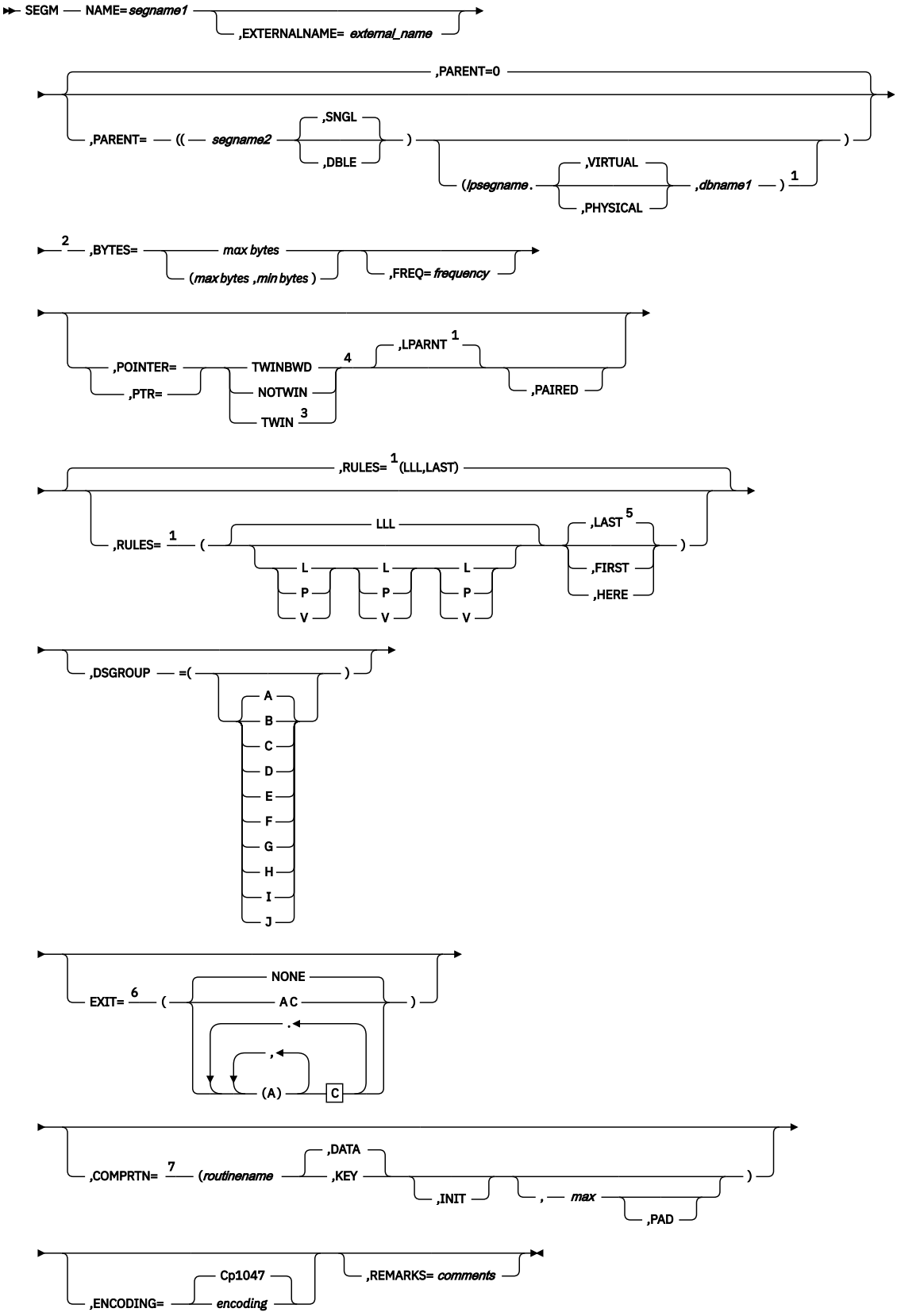
[5] Used for the Data Capture exit routine. You can specify more than one exit routine on a SEGM statement.

[6] Used for Segment Edit/Compression exit routine.

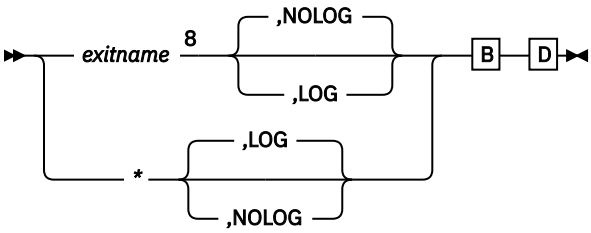
[7] If an exit routine name is not required because only logging is requested, specify the exit name as an asterisk (*). The default logging parameter in this case is LOG.

[8] Used to control the CASCADE options. If any (CASCADE B) suboptions are specified with NOCASCADE in an **EXIT=** parameter, they will be ignored, and the value for those suboptions will be set to NO*xxxx*.

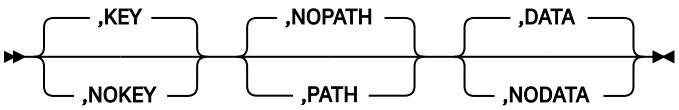
PHIDAM database SEGM statement


SEGM
NAME=segname1
,EXTERNALNAME= external_name
,PARENT=0
,PARENT=
((
segname2
,SNGL
,DBLE
)
(lpsegname.
,VIRTUAL
,PHYSICAL
,dbname1
) 1
) 2
,BYTES=
max bytes
(max bytes ,min bytes)
,FREQ= frequency
,POINTER=
,PTR=
TWINBWD
NOTWIN
TWIN 3
4
,LPARNT 1
,PAIRED
,RULES= 1 (LLL,LAST)
,RULES= 1 (
LLL
L
P
V
,LAST 5
,FIRST
,HERE
)
,DSGROUP =(
A
B
C
D
E
F
G
H
I
J
)
EXIT= 6 (
NONE
AC
(A)
C
)
,COMPRTN= 7 (routinename
,DATA
,KEY
,INIT
, max
,PAD
)
,ENCODING=
Cp1047
encoding
,REMARKS= comments

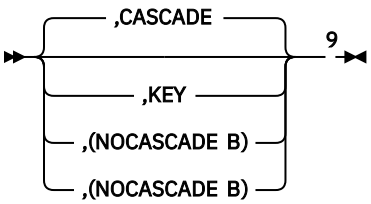
A



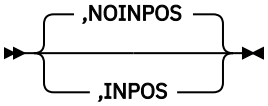
B



C



D



Notes:

[1] Optional for PHIDAM logical relationships.

[2] The PARENT= keyword can be omitted, or PARENT=0 can be specified for the root segment type of a database.

[3] TWIN is not allowed for the root segment.

[4] Required for PHIDAM logical relationships; otherwise, it is optional.

[5] Required when a segment type does not have a unique sequence field. LAST is the default.

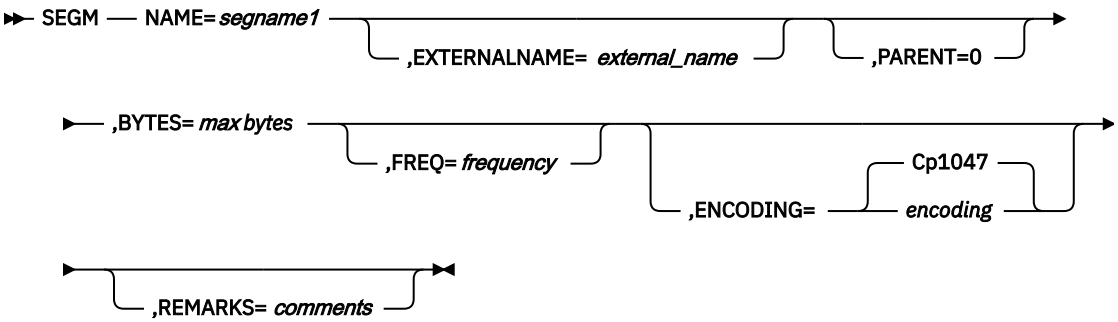
[6] Used for the Data Capture exit routine. You can specify more than one exit routine on a SEGM statement.

[7] Used for Segment Edit/Compression exit routine.

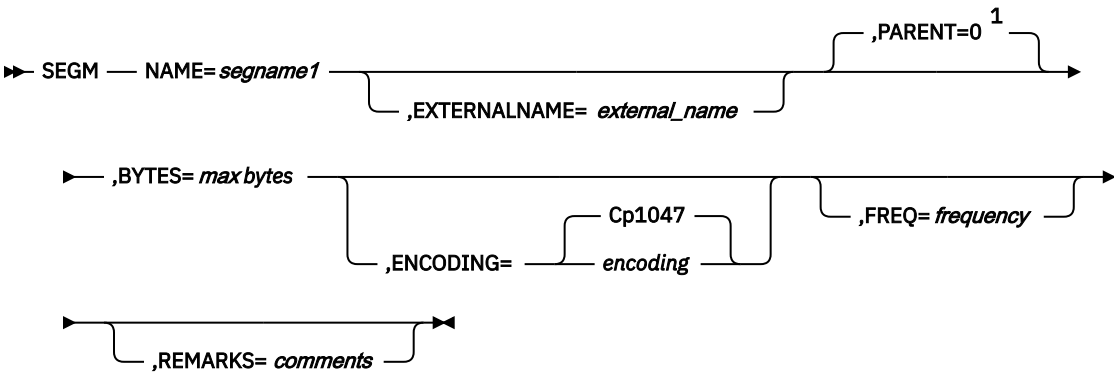
[8] If an exit routine name is not required because only logging is requested, specify the exit name as an asterisk (*). The default logging parameter in this case is LOG.

[9] Used to control the CASCADE options. If any (CASCADE B) suboptions are specified with NOCASCADE in an **EXIT=** parameter, they will be ignored, and the value for those suboptions will be set to NO*xxxx*.

PSINDEX database SEGM statement



SHSAM database SEGM statement



Notes:

[1] The PARENT= keyword can be omitted, or PARENT=0 can be specified for the root segment type of a database.

SEGM statement parameter description

For the SEGM statement, you can use the following abbreviations in place of keywords specified in the macro definitions:

Table 5. Keyword abbreviations

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

SEGM
Identifies this statement as a segment definition statement.

NAME=
Specifies the name of the segment type being defined.

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

Restriction: The first character of the name cannot be numeric.

GSAM databases do not support the NAME= parameter on the SEGM statement.

PARENT=
Specifies the names of the physical and logical parents of the segment type being defined, if any.

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

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

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

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

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

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

VIRTUAL | PHYSICAL
Specifies whether the concatenated key of the logical parent (LPCK) is stored as a part of the logical child segment. Specify the parameter only for logical child segments. If PHYSICAL is specified, the LPCK is stored with each logical child segment. If VIRTUAL is specified, the LPCK is not stored in the logical child segment. PHYSICAL must be specified for a logical child segment whose logical parent is in a HISAM database. It must be specified also for a logical child segment that is sequenced on its physical twin chain through use of any part of the concatenated key of the logical parent.

- PHDAM and PHIDAM
  - PHYSICAL is the default for PHDAM and PHIDAM.
  - If VIRTUAL is specified for PHDAM or PHIDAM, it is ignored, and PHYSICAL is used.
- HDAM and HIDAM
  - VIRTUAL is the default for HDAM and HIDAM.
  - Symbolic pointers in HDAM and HIDAM databases use the LPCK and require the PHYSICAL specification.

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

BYTES=
Specifies the length of the data portion of a segment type in bytes using unsigned decimal integers. This parameter is required. For segments that are logical children, this length includes the

concatenated key of the logical parent when either VIRTUAL or PHYSICAL is specified or defaulted to in the PARENT parameter.

***maxbytes* and *minbytes* in fixed-length segments**
For fixed-length segments, the *maxbytes* parameter specifies the amount of storage used for the data portion of the segment. The *minbytes* parameter cannot be specified for a fixed-length segment, including a fixed-length compressed segment. The maximum length specified for a segment type must not exceed the maximum record length of the storage device used minus any prefix or record overhead.

For VSAM, the maximum record length is 30713 bytes; for tape, the maximum is 32760 bytes. The minimum length that can be specified for *maxbytes* must be large enough to contain all fields defined for the segment type. If the segment is a logical child segment type, the length must be sufficient to contain the concatenated key of the logical parent.

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

***maxbytes* and *minbytes* in variable-length segments**
Defines a segment type as variable-length if the *minbytes* parameter is included. The *maxbytes* field specifies the maximum length of any occurrence of this segment type. The maximum and minimum allowable values for the *maxbytes* parameter are the same values as described for a fixed-length segment.

If the segment is processed by a compression routine, set the *maxbytes* field to accommodate control information to indicate whether the segment length can be longer than the specified maximum definition in order to avoid an abend 0799. To allow for the expansion, add an arbitrary value of 10 bytes to the *maxbytes*.

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

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

Because segments in an HSAM, SHSAM, or SHISAM database cannot be variable-length, the *minbytes* parameter is invalid for these databases.

In a Fast Path DEDB, a segment starts with a 2-byte field, which defines the length of the segment including the 2-byte length field, followed by user data specified by a FIELD statement. The value of *minbytes* can be specified from a minimum of 4 bytes to a maximum of *maxbytes*; however, the *minbytes* value must be large enough to contain this segment's sequence field (that is, *minbytes* ≥ START - 1 + BYTES of the sequence field following the SEGM statement). For example, the smallest *minbytes* value for a segment with a 20-byte sequence field length and START = 7 is 26. On any given DL/I call, the actual segment length can fall anywhere between a length that includes the sequence field and the value of *maxbytes*. The value of *maxbytes* must not exceed the control interval size minus 120.

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

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

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

FREQ=

Specifies the estimated number of times that this segment is likely to occur for each occurrence of its physical parent. Value must be a whole, unsigned decimal number from 1 to 16777215. If this segment is a root segment, "frequency" is the estimate of the maximum number of database records that appear in the database that is being defined. The value of the FREQ= parameter when applied to dependent segments is used to determine the logical record length and physical storage block sizes for each data set group of the database.

The IF0110 ARITHMETIC OVERFLOW or IEV103 MULTIPLICATION OVERFLOW assembler error message can occur when the DBDGEN utility is attempting to calculate a recommended logical record length. If this error occurs during an HSAM, SHSAM, or HISAM DBD generation, you might want to determine the logical record length and physical block size.

FREQ= is not valid for segments in Fast Path DEDB or MSDB databases.

INPOS|NOINPOS

An optional parameter to capture next twin data on an ISRT. The default is NOINPOS. The twin data of the twin following an inserted segment is captured if INPOS is specified and the following conditions are true:

- An ISRT of a non-unique segment is made.
- An ISRT rule of HERE is used.

If the new segment is the only twin instance or last in the twin chain, no twin data exists.

POINTER=

Specifies the pointer fields to be reserved in the prefix area of occurrences of the segment type that is being defined. These fields are used to relate this segment to its immediate parent segments and twin segments.

The use of the POINTER= parameter is primarily for HDAM, HIDAM, PHDAM, and PHIDAM databases. In addition, it can be used for segment types that are defined in HISAM databases that participate in logical relationships with segment types in HDAM or HIDAM databases.

Important: If a segment type is being defined in an HSAM or SHSAM database, the POINTER= parameter must be omitted. If the segment type that is being defined is in a HISAM database and does not participate in a logical relationship, omit the POINTER= parameter.

The following list describes some general attributes of the keyword options:

- Selected keyword options can be specified in any order, and must be separated by commas.
- A keyword option can be specified only once.
- All keywords are optional.
- One keyword option can be selected from each line.
- A keyword option or its abbreviation can be selected:

Table 6. POINTER= keywords and abbreviations

Keyword option	Abbreviation
HIER	H
HIERBWD	HB
TWIN	T
TWINBWD	TB
NOTWIN	NT
LTWIN	LT
LTWINBWD	LTB

Table 6. POINTER= keywords and abbreviations (continued)

Keyword option	Abbreviation
PAIRED	
LPARNT	LP
CTR	C

The keyword options of the POINTER= parameter have the following meanings:

HIER [H]
Reserves a 4-byte hierarchic forward pointer field in the prefix of occurrences of the segment type being defined. HALDB does not support HIER.

HIERBWD [HB]
Reserves a 4-byte hierarchic forward pointer field and a 4-byte hierarchic backward pointer field in the prefix of occurrences of the segment type being defined. Hierarchic backward pointers provide increased delete performance. HALDB does not support HIERBWD.

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

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

Recommendation: This option is recommended for HIDAM and PHIDAM database root segments.

NOTWIN [NT]
Prevents space from being reserved for a physical twin forward pointer in the prefix of occurrences of the segment type being defined.

NOTWIN can be specified for a dependent segment type if:

- The physical parent does not have hierarchic pointers specified.
- No more than one occurrence of the dependent segment type is stored as a physical child of any occurrence of the physical parent segment type.

In addition, NOTWIN can be specified for the root segment type of HDAM and PHIDAM databases, but only when the randomizing module does not produce synonyms (keys with different values having the same block and anchor point).

When NOTWIN is specified for a dependent segment type and an attempt is made to load or insert a second occurrence of the dependent segment as a physical child of a given physical parent segment:

- An LB status code is returned when trying to insert the second occurrence during initial load.
- An II status code is returned when trying to insert the second occurrence after initial load.

Any attempt to load or insert a synonym is rejected with an LB or II status code.

LTWIN [LT]
Is used for virtually paired logical relationships only when defining a real logical child. Reserves a 4-byte logical twin forward pointer field in the prefix of occurrences of the logical child segment type being defined. This parameter can only be specified if the segment type being defined is a logical child and is being defined in an HDAM or HIDAM database. If PAIRED is specified, the LTWIN parameter is invalid. HALDB does not support LTWIN.

LTWINBWD [LTB]
Is used for virtually paired logical relationships only when defining a real logical child. Reserves a 4-byte logical twin forward pointer field and a 4-byte logical twin backward field in the prefix of occurrences of the logical child segment type being defined. This parameter can only be specified

if the segment being defined is a logical child and is being defined in an HDAM or HIDAM database. If PAIRED is specified, the LTWIN parameter is invalid. HALDB does not support LTWINBWD.

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

LPARNT [LP]
This parameter can be specified only when the segment type that is being defined is a logical child and the logical parent is in an HDAM, HIDAM, PHDAM, or PHIDAM database. If the logical parent is in a HISAM database, omit this parameter and specify PHYSICAL in the PARENT= parameter for the segment that is being defined.

For HDAM, HIDAM, and HISAM databases, LPARNT reserves a 4-byte logical parent pointer field in the prefix of occurrences of the segment type being defined.

For PHDAM and PHIDAM databases, LPARNT reserves a 28-byte extended pointer set in the prefix of occurrences of the segment type being defined.

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

PAIRED
Indicates that this segment participates in a bidirectional logical relationship. This parameter is specified for the following types:

- A virtual logical child segment type
- Both physically paired logical child segment types in a bidirectional logical relationship

If PAIRED is specified, the LTWIN and LTWINBWD parameters are invalid.

POINTER= Parameter Default Values
The default option for the POINTER= parameter in any HIDAM or HDAM DBD is:

```
PTR=(TWIN,LTWIN,LPARNT)
```

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

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

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

If the POINTER= parameter is explicitly stated on a SEGM statement, the segment contains the pointers specified and any pointers that are required by IMS for correct operation. For example, LTWIN and LPARNT pointers are created as required. The default values are only used when the parameter is omitted entirely. The following table illustrates use of the POINTER= parameter parameters for various types of DBD generations.

Table 7. Use of POINTER= parameters (no logical relationship)

			Segment definition				
			Physical segments contained in database type				
Purpose	**Keyword parameter**	**Logical segments GSAM MSDB DEDB**	**HSAM SHSAM SHISAM**	**HISAM**	**HDAM HIDAM**	**PHDAM PHIDAM**	**INDEX PSINDEX**
Pointer to next segment in hierarchy	HIER	INVALID	VALID	IGN	VALID	IGN	IGN
Pointer to next and previous segments in hierarchy	HIERBWD	INVALID	INVALID	IGN	VALID	IGN	IGN
Pointer to next occurrence of physical twins	TWIN	INVALID	INVALID	IGN	VALID	VALID	IGN
Pointer to next and previous occurrence of physical twins	TWINBWD	INVALID	INVALID	IGN	VALID	VALID	IGN
Counter field in prefix	CTR	INVALID	INVALID	VALID	VALID	IGN	IGN
Pointer to next occurrence of logical twin	LTWIN	INVALID	INVALID	IGN	VALID[1]	IGN	IGN
Pointer to next and previous occurrence of logical twins	LTWINBWD	INVALID	INVALID	IGN	VALID[1]	IGN	IGN
Pointer to logical parent segment	LPARNT	INVALID	INVALID	VALID[2]	VALID[3]	VALID[3]	IGN
Logical relationship between HS-HS or HS-HD or HD-HD	PAIRED	INVALID	INVALID	VALID[4]	VALID[5]	VALID[5]	IGN

Key:

- INVALID — This parameter cannot be specified.
- IGN — This parameter can be specified but it is ignored.
- VALID — This parameter is valid and used as indicated in the following notes.

Table 7. Use of POINTER= parameters (no logical relationship) (continued)

			Segment definition				
			Physical segments contained in database type				
Purpose	**Keyword parameter**	**Logical segments GSAM MSDB DEDB**	**HSAM SHSAM SHISAM**	**HISAM**	**HDAM HIDAM**	**PHDAM PHIDAM**	**INDEX PSINDEX**

Notes:

1. Used when a logical child segment being defined participates in a logical relationship. This should be specified if the segment exists within HDAM, HIDAM, PHDAM or PHIDAM, and the logical parent relates to the logical child with direct addresses (logical child pointers).
2. Can be used when a logical child segment is being defined in a HISAM database and the logical parent is defined in an HDAM, HIDAM, PHDAM, or PHIDAM database.
3. Can be used when a logical child segment is being defined in an HDAM, HIDAM, PHDAM, or PHIDAM database and the logical parent is in an HDAM, HIDAM, PHDAM, or PHIDAM database.
4. Can be used when a logical child segment is being defined in a HISAM database and the logical parent is defined in a HISAM, HDAM, HIDAM, PHDAM, or PHIDAM database, and the logical relationship is bidirectional.
5. Used when a bidirectional logical relationship is being defined with two logical child segments, both physically present or on the SEGM statement for a virtual logical child.

RULES=

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

path type values

Specifies the path type that must be used to insert, delete, or replace a segment.

The first column applies to segment insertion, the second column applies to segment deletion, and the third column applies to segment replacement. Each of the three columns can contain the same or different characters, but you must select a value from each column for a total of three values. These parameters are specified for logical child segments and for their physical and logical parent segments. They should be omitted for all segment types that do not participate in logical relationships. The values are: P specifies physical, L specifies logical, V specifies virtual, and B specifies bidirectional virtual.

FIRST | LAST | HERE

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

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

LAST is the default except for DEDB segments.

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

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

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

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

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

DSGROUP=
Specifies multiple data set groups for PHDAM and PHIDAM databases. The format is DSGROUP=c, where c is equivalent to the letters A through J. This enables you to divide PHDAM and PHIDAM databases into a maximum of ten data set groups. The default for every segment is A (single set for data per partition). If specified on the root segment, it must be DSGROUP=A.

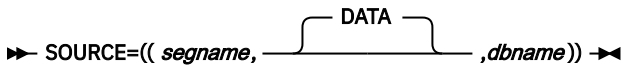
Restriction: Gaps in the A-J sequence are not allowed. For example, if DSGROUP=C is specified on a SEGM statement, there must also be at least one SEGM statement with DSGROUP=B, and each HALDB partition will have A, B, and C data sets.

SOURCE=
Is used for two purposes:

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

Restriction: The SOURCE keyword is not allowed for PHDAM and PHIDAM databases because they support only physical pairing.

When defining a virtual logical child the statement is:

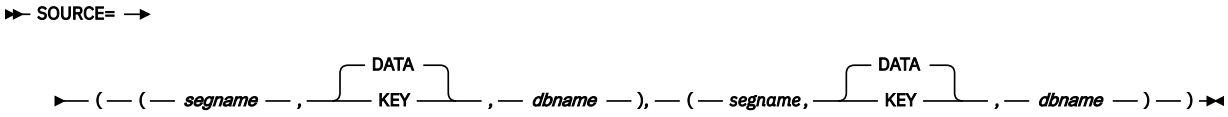


segname
Specifies the name of the real, logical child

DATA
Indicates that both the key and the data portions of *segname* are to be used in constructing the segment. This parameter is required.

dbname
Specifies the name of the physical database that contains the real logical child.

When defining a segment type in a logical database the statement is:



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

segname
Is the name of the segment type in the physical database.

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

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

dbname
Specifies the name of the physical database that contains *segname*. The second occurrence of (*segname*, KEY|DATA, *dbname*) refers to the logical or physical parent segment type in a physical database that is used for the destination parent part of a concatenated segment in this logical database. The description of each parameter for the second occurrence is the same as described for the first occurrence.

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

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

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

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

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

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

EXIT=
Specifies that the Data Capture exit routine is used. You can specify multiple exit routine names on a single SEGM statement. You can select different data options for each exit routine. The order you list the exit routines within the parameter determines the order the exit routines are called.

When specified on the SEGM statement, the EXIT= parameter can either override the specification on the DBD or limit the parameter to specific segments. The EXIT= parameter applies only to the particular segments within the physical database specified. However, when applied to logical children segments, the exit routine must be specified on the real logical child, not the virtual logical child. The following physical databases are supported by this exit routine:

- HDAM
- HIDAM
- PHDAM
- PHIDAM
- HISAM
- SHISAM
- DEDB

If the exit routine is not specified for a supported database organization or a supported segment type, DBDGEN fails.

If the job name of a CCTL or ODBM address space is specified on the **SUPPDCAPNAME=** parameter, which is in the DATABASE section of the DFSDF*xxx* member of the IMS PROCLIB data set, the exit routine is not called for data updates invoked by the specified job, even if a Data Capture exit routine is specified on the EXIT= parameter.

The EXIT= parameter can also be specified on the DBD statement.

exit_name
Specifies the name of the exit routine that processes the data. This parameter is required. The name must follow standard naming conventions. A maximum of 8 alphanumeric characters is allowed. You can specify an asterisk (*) instead of an exit routine name to indicate that you want logging only. If this is done, the logging parameter default is LOG. If you do specify an exit routine, the logging parameter is NOLOG.

The following operands are optional.

NONE
Nullifies an exit routine specified on the DBD statement. It must be specified on the SEGM statement to indicate the DBD exit name does not apply to that specific segment.

EXIT=NONE explicitly nullifies the exit specified on the DBD for virtual logical children.

LOG | NOLOG
Specifies whether data capture control blocks and data are to be written to the IMS system log.

LOG
Requests that the data capture control blocks and data be written to the IMS system log.

For more information, see Asynchronous data propagation (System Programming APIs).

NOLOG
Indicates that no data capture control blocks or data is written to the IMS system log.

KEY | NOKEY
Specifies whether the physical concatenated key is passed to the exit routine.

KEY
Specifies that the exit routine is passed the physical concatenated key. This key identifies the physical segment updated by the application.

KEY is the default.

NOKEY
Specifies the physical concatenated key is not required for the exit routine.

DATA | NODATA
Specifies whether physical segment data is to be passed to the Data Capture exit routine for updating.

DATA
Passes physical segment data to the Data Capture exit routine for updating. When DATA is specified and a Segment Edit/Compression exit routine is also being used, the data passed is expanded data.

DATA is the default.

NODATA
Can be specified when the exit routine does not require segment data. Use NODATA to avoid the overhead created from saving physical segment data.

NOPATH | PATH
Specifies whether the exit routine requires data from segments in the physical root's hierarchical path.

NOPATH
Indicates that the exit routine does not require data from segments in the physical root's hierarchical path. NOPATH is an efficient way to avoid the processing time needed to retrieve path data.

NOPATH is the default.

PATH
Can be specified when the data from each segment in the physical root's hierarchic path must be passed to the exit routine for an updated segment. Use PATH to allow an application to separately access several segments for insertion, replacement, or deletion.

You can use the PATH option when information from segments in the path is needed to compose the Db2 for z/OS primary key. The Db2 for z/OS primary key would then be used in a propagation request for a dependent segment update. Typically, you need this kind of segment information when the parent contains the key information and the dependent contains additional data that would not fit in the parent segment.

You can also use PATH when additional processing is necessary. It could be that you are not accessing several segments with one call; for example, you did not invoke the D command code. In this case, additional processing is necessary if the application is to access each segment with a separate call.

DLET | NODLET

Specifies whether X'99' log records are written for DLET calls.

Note: DLET or NODLET can only be specified for a DEDB.

If you specify this parameter in the SEGM statement, it overrides the specification for the DBD statement.

DLET
X'99' log records are written for DLET calls.

DLET is the default.

NODLET
No X'99' log records are written for DLET calls.

BEFORE | NOBEFORE

Specifies whether the before data is included in X'99' log records for REPL calls.

Note: BEFORE or NOBEFORE can only be specified for a DEDB.

If you specify this parameter in the SEGM statement, it overrides the specification for the DBD statement.

BEFORE
Before data is included in X'99' log records for REPL calls.

BEFORE is the default.

NOBEFORE
No before data is included in X'99' log records for REPL calls.

CASCADE | NOCASCADE
Specifies whether the exit routine is called when DL/I deletes this segment.

CASCADE
Indicates that the exit routine is called when DL/I deletes this segment because the application deleted a parent segment. Using CASCADE ensures that data is captured for the defined segment.

CASCADE is the default.

The CASCADE parameter has three suboptions. These suboptions control the way data is passed to the exit routine. If you specify suboptions, you must enclose the CASCADE parameter and the suboptions within parentheses.

KEY | NOKEY
Specifies whether the physical concatenated key is passed to the exit routine.

KEY
Passes the physical concatenated key to the exit routine. This key identifies the segment being deleted by a cascade delete.

KEY is the default.

NOKEY
Can be used when the exit routine does not require the physical concatenated key of the segment being deleted.

DATA | NODATA
Specifies whether segment data is to be passed to the exit routine.

DATA
Passes segment data to the exit routine for a cascade delete. DATA also identifies the segment being deleted when the physical concatenated key is unable to do so.

DATA is the default.

NODATA
Can be specified when the exit routine does not require segment data. NODATA reduces the significant storage and performance requirements that result from saving physical segment data.

NOPATH | PATH
Specifies whether the exit routine requires segment data in the physical root's hierarchical path.

NOPATH
Indicates the exit routine does not require segment data in the physical root's hierarchical path. Use NOPATH to eliminate the substantial amount of path data needed for a cascade delete.

NOPATH is the default.

PATH
Can be specified to allow an application to separately access several segments for a cascade delete.

NOCASCADE
Indicates that the exit routine is not called when DL/I deletes this segment. Cascade delete is not necessary when a segment without dependents is deleted.

Note: If any (CASCADE B) suboptions are specified with NOCASCADE in an **EXIT=** parameter, they will be ignored, and the value for those suboptions will be set to NO*xxxx*.

NOSSPCMD | SSPCMD

An optional parameter that indicates whether command codes related to Fast Path subset pointers (SSP) be captured. The default is NOSSPCMD. It is recommended that this option be specified only on segments that involve subset pointers.

The following table indicates which command codes are captured for a given DL/I call:

Table 8. Command codes that are captured for DL/I calls

DL/I call	Details
G* (get calls)	M, S, W, Z. R is captured if at least one of M, S, W, or Z is also on the same SSA, or along with the PATH data if PATH is requested.
REPL	M, S, W, Z
DLET	Z
ISRT	M, S, W, Z. R is captured if the segment is being inserted or if it was specified on an SSA of a segment not being inserted but PATH data is requested.

NOINPOS | INPOS

An optional parameter to request that the next twin data be captured on an ISRT call. The default is NOINPOS. The twin data of the twin that follows an inserted segment will be captured if INPOS is specified and the following conditions are true:

- An ISRT of a non-unique segment is made.
- An ISRT rule of HERE is used.

If the new segment is the only twin instance or last in the twin chain, no twin data will be captured.

NOFLD | FLD

An optional parameter to request that updates that are made by a DEDB FLD call be captured. This option is valid only for a DEDB. The information captured is logged only in the X'9904' log records if option LOG is specified. It is not passed to the Data Capture exit routine.

COMPRTN=

Selects a Segment Edit/Compression exit routine for either DEDB or full-function database.

For segment edit/compression of full-function database

Do not specify this keyword if the SOURCE keyword is used. The DL/I COMPRTN keyword is invalid during DBDGEN for MSDB, HSAM, SHSAM, SHISAM, INDEX, and logical databases. It is also invalid for logical child segments in any database. When used for a HISAM database, it must not change the sequence field offset for HISAM root segments. In addition, the minimum segment length that can be specified for a segment type where the segment edit/compression option is specified is 4 bytes.

Remember: If you are using a segment edit/compression exit routine and defined your segments as variable-length, be aware that when a variable-length segment is compressed, it is padded with null bytes up to the minimum segment length that was defined in the DBD. Minimum segment length essentially overrides the compression; this enables you to provide additional space during load time for segments that are heavily compressed.

routinename

Specifies the name of the user-supplied edit/compression exit routine. This name must be a 1- to 8-character alphanumeric value, must not be the same as any other name in IMS.SDFSRESL, and must not be the same as DBDNAME.

DATA

Specifies that the indicated exit routine condenses or modifies data fields only. Sequence fields must not be modified, nor data fields that change the position of the sequence field in respect

to the start of the segment. DATA is the default value if a compression routine is named but no parameter is selected.

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

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

max
Specifies the maximum number of bytes by which fixed-length segments can increase during compression exits. You can specify from 1 to 32 767 bytes. The default for *max* is 10.

PAD
Indicates that the numeric value supplied by MAX should be used for padding and not for MAX. The numeric range of 1 to 32 767 indicates a size to which an inserted segment will be padded when the compression of that segment results in a length somewhat less than the PAD value.

For segment edit/compression of DEDB

routinename
Specifies the z/OS load module name of the user-supplied segment edit/compression exit routine. The routine name is required.

DATA
Specifies that only the user data part of the segment is compressed. DATA is the default.

Restriction: The KEY parameter is not supported for DEDB. If you specify the KEY parameter, an error message is issued and DBDGEN is terminated.

INIT
Allows the segment compression exit routine to gain control immediately after the first area in the database is opened and returns control immediately before the last area in the database is closed. As long as the segment length is within the values specified by DBDGEN, no errors occur while checking the field qualification for segment compression or expansion.

Restriction: The COMPRTN= keyword is prohibited on DEDB segments containing a unique key field located at the end of the segment. If you use COMPRTN= to process these types of segments, DBDGEN fails and message DGEN440 is issued.

ENCODING=
An optional 1- to 25-character field that specifies the encoding of the character data in the segment.

The value specified on the ENCODING keyword cannot contain the following characters:

- Single and double quotation marks
- Blanks
- Less than (<) and greater than (>) symbols
- Ampersands (&)

The value of the ENCODING parameter in the SEGM statement overrides the value of the ENCODING parameter in the DBD statement for this segment. If the ENCODING parameter is not specified on the SEGM statement, the default value is either the value of the ENCODING parameter on the DBD statement or, if ENCODING was not specified on the DBD statement, the value Cp1047, which specifies EBCDIC encoding.

This value can be overridden in individual fields by the ENCODING parameter in the DFSMARSH statement.

EXTERNALNAME=
An optional alias for the NAME= parameter. Java™ application programs use the external name to refer to the segment.

GSAM databases require the EXTERNALNAME parameter on the SEGM statement.

Specify an external name as a 1- to 128-character uppercase alphanumeric string. An external name can include underscore characters.

The external names specified on the SEGM statement must be unique within a DBD.

The default value of the EXTERNALNAME parameter is the value of the NAME parameter.

Restriction: External names cannot be reserved SQL keywords or begin with DFS.

If the EXTERNALNAME parameter is not specified and a reserved SQL keyword is specified in the NAME parameter, EXTERNALNAME accepts the NAME value as the default external name after appending "_TBL" to the NAME value.

REMARKS=
Optional user comments. A 1- to 256-character field.

If your comments contain special characters, such as commas or blank spaces, enclose the full comment string in single quotation marks.

The value specified on the REMARKS keyword cannot contain the following characters:

- Single quotation marks, except when they are used to enclose the full comment string. If a single quotation mark is entered before the end of the full comment string, the remainder of the comment string is truncated. The following examples show correct and incorrect usages of single quotation marks on the REMARKS keyword:

  CORRECT
  ```
  REMARKS='These remarks apply to the XYZ application'
  ```
  INCORRECT
  ```
  REMARKS='These remarks apply to the 'XYZ' application'
  ```
- Double quotation marks.
- Less than (<) symbols.
- Greater than (>) symbols.
- Ampersands (&).

Related concepts
Data Capture exit routines (Database Administration)
Origin of GSAM data set characteristics (Application Programming)
Related reference
“DATASET statements” on page 48
A DATASET statement defines a data set group within a database.
Segment edit/compression exit routines (Exit Routines)
Portable SQL keywords restricted by the IMS Universal JDBC drivers (Application Programming)
Related information
0799 (Messages and Codes)

LCHILD statements

The LCHILD statement defines a logical relationship between two segment types in a DEDB, HISAM, HIDAM, HDAM, PHDAM, or PHIDAM database or a logical relationship between a segment type in any two of these databases.

Restriction: Do not specify an LCHILD statement for the primary index of a PHIDAM database.

Logical relationships

Following any SEGM statement that defines a logical parent segment type in a DBDGEN input deck, there must be one LCHILD statement for each segment type that is a logical child of that logical parent, except

for virtual logical child segment types. These LCHILD statements establish the relationships between the logical parent and its logical child segment types. The SOURCE= parameter of a SEGM statement that defines a virtual logical child segment type establishes the same relationship between a logical parent and a virtual logical child segment type.

HIDAM primary index relationship

Two LCHILD statements are used to establish the index relationship required between the HIDAM primary index database and the root segment type of a HIDAM database.

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

Secondary index relationships

Two LCHILD statements are used to establish each secondary index relationship.

Following a SEGM statement that defines an index target segment type, there must be one LCHILD statement for each index pointer segment type that points to that index target segment type. Each LCHILD statement following the SEGM for an index target segment type identifies the index pointer segment type that points to the index target.

Fast Path DBDs support multiple LCHILD statements under a single SEGM statement. You can specify as many LCHILD statements in the DEDB DBD as there are search fields of equal lengths from each source segment to form multiple secondary index pointer segments that point to a single secondary index.

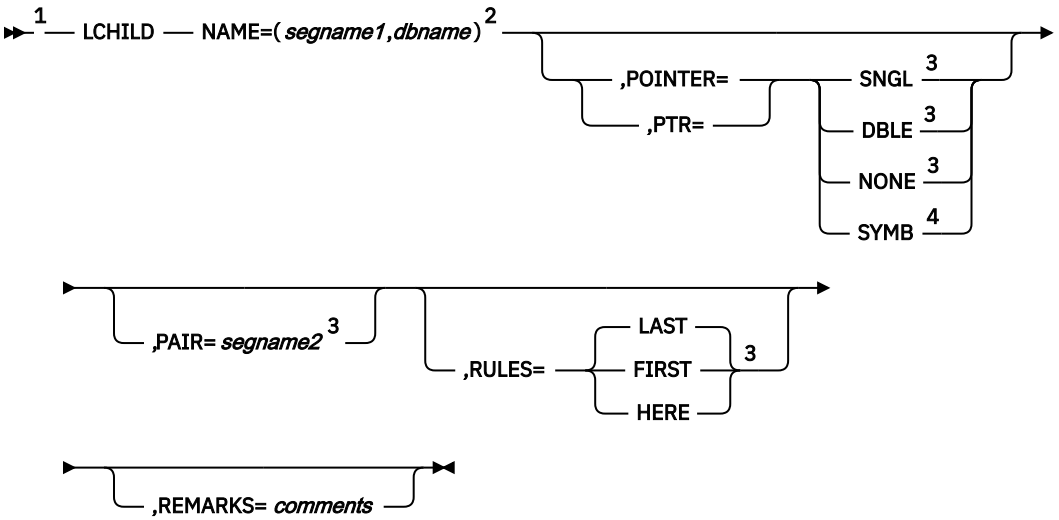
To define multiple secondary index segments with the same segment name for a single target segment from a single source segment, define two or more LCHILD/XDFLD statement pairs under the SEGM statement of a target segment.

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

Fast Path secondary indexes do not support PAIR and RULES operands on LCHILD statements. The PAIR= and RULES= parameters on an LCHILD statement are used for logical relationships and are invalid parameters on an LCHILD statement in a primary DEDB database DBD.

The format of the LCHILD statement for each database type is shown in the following examples.

HISAM database LCHILD statement



Notes:

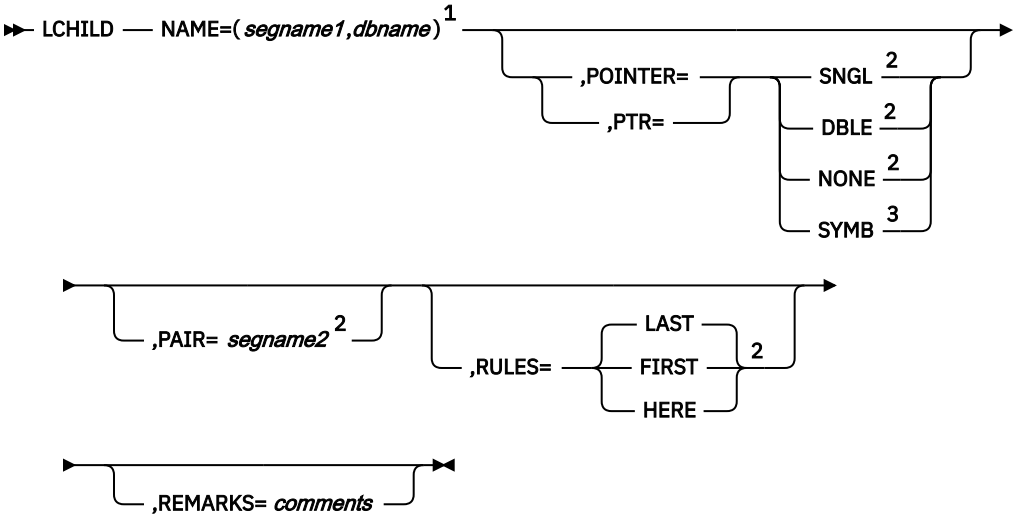
[1] If a HISAM secondary index database or a SHISAM secondary index database has two or more user partition databases, specify two or more user partition secondary index database names in the NAME= parameter.

[2] Used for logical relationships or secondary indexing.

[3] Used for logical relationships.

[4] If symbolic pointing is specified for the index target segment type when defining its physical database, specify symbolic pointing in the secondary index for that segment type. If SYMB is specified for the target segment of a secondary index, the PTR=SYMB is specified on the LCHILD statement of the INDEX DBD also. For Fast Path secondary indexing, the PTR=SYMB parameter must be explicitly specified on an LCHILD statement because Fast Path secondary indexing only supports symbolic pointers, and PTR=SNGL is the default.

HDAM database LCHILD statements



Notes:

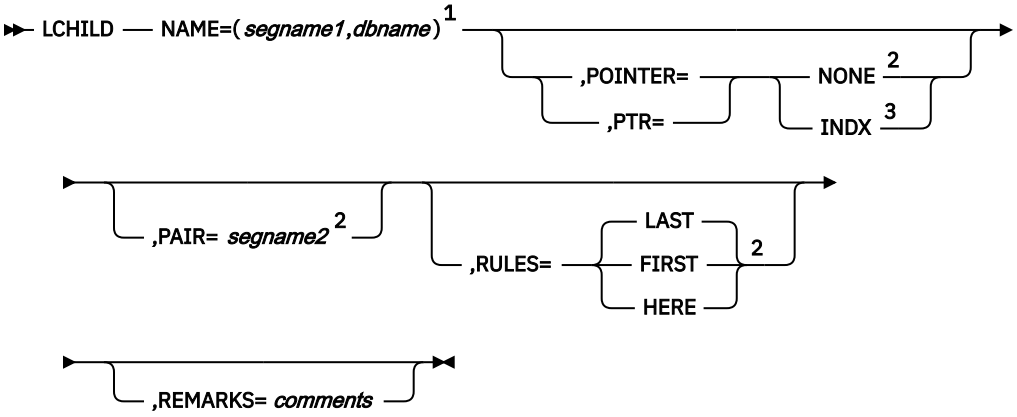
[1] Used for logical relationships or secondary indexing.

[2] Used for HDAM, HISAM, and HIDAM logical relationships.

[3] If symbolic pointing is specified for the index target segment type when defining its physical database, specify symbolic pointing in the secondary index for that segment type. If SYMB is specified

for the target segment of a secondary index, the PTR=SYMB is specified on the LCHILD statement of the INDEX DBD also.

PHDAM database LCHILD statements



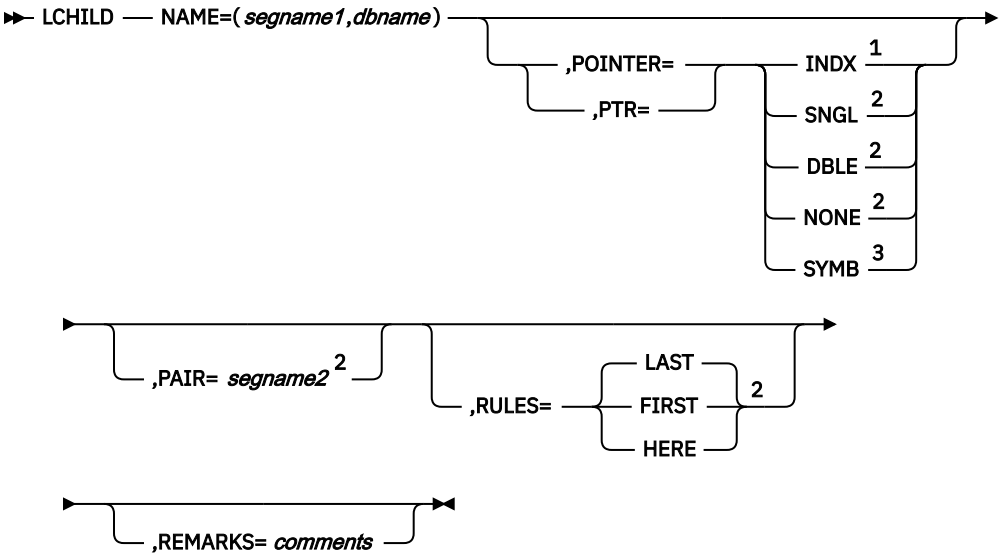
Notes:

1 Used for logical relationships or secondary indexing.

2 Used for HDAM, HISAM, and HIDAM logical relationships.

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

HIDAM Database LCHILD Statements



Notes:

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

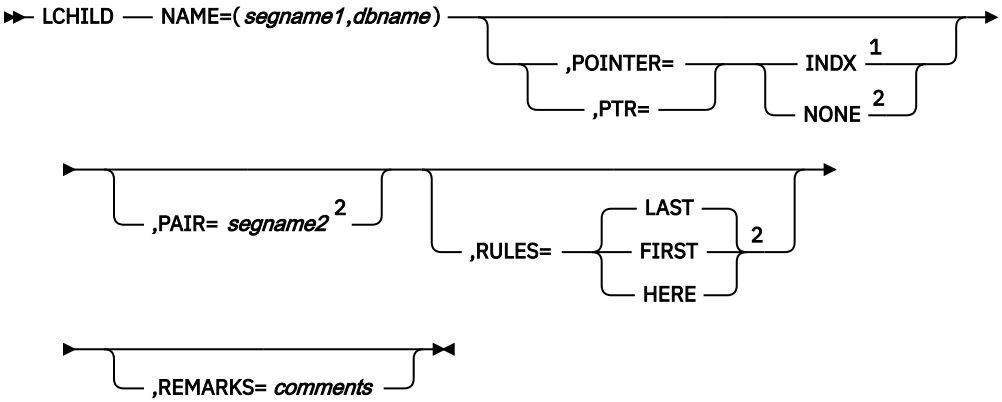
2 Used for HDAM, HISAM, and HIDAM logical relationships.

3 If symbolic pointing is specified for the index target segment type when defining its physical database, specify symbolic pointing in the secondary index for that segment type. If SYMB is specified

for the target segment of a secondary index, the PTR=SYMB is specified on the LCHILD statement of the INDEX DBD also.

PHIDAM Database LCHILD Statements

Restriction: Do not enter an LCHILD statement for the primary index of a PHIDAM database.

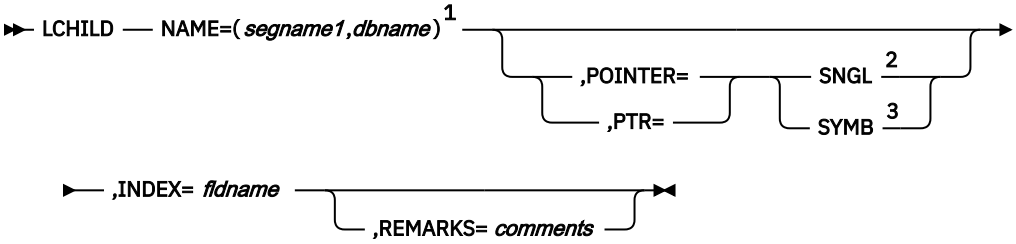


Notes:

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

[2] Used for HDAM, HISAM, and HIDAM logical relationships.

INDEX database LCHILD Statement for full-function secondary index databases



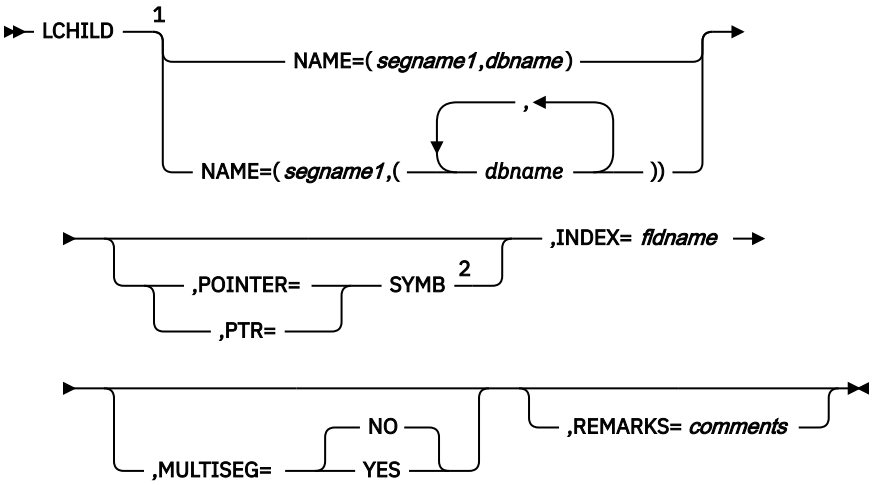
Notes:

[1] Primary indexing and secondary indexing.

[2] Required for primary index of HIDAM database.

[3] If symbolic pointing is specified for the index target segment type when defining its physical database, specify symbolic pointing in the secondary index for that segment type. If SYMB is specified for the target segment of a secondary index, the PTR=SYMB is specified on the LCHILD statement of the INDEX DBD also.

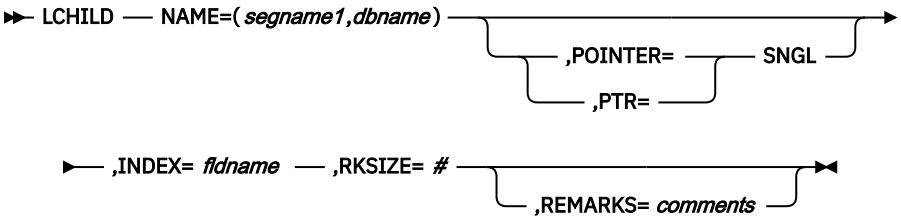
INDEX database LCHILD Statement for DEDB secondary index databases



Notes:

[1] Primary indexing and secondary indexing.
[2] Fast Path secondary indexes require symbolic pointers.

PSINDEX database LCHILD statement



LCHILD statement parameter description

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

Keyword
Abbreviation

POINTER
PTR

FIRST
F

LAST
L

HERE
H

NAME=
The *segname1* parameter specifies the name of the logical child, index pointer, index target, HIDAM or PHIDAM root segment type that is to be associated with the segment type defined by the preceding SEGM statement in the DBD generation input deck. The *dbname* parameter is the name of the database that contains the segment type specified in *segname1*. *dbname* can be omitted when *segname1* is defined in this DBD generation. Both *segname1* and *dbname* must be one- to eight-character alphanumeric values.

POINTER=
Specifies the pointers used in logical or index relationships. When the POINTER= keyword is omitted from any index DBD generation, POINTER=SNGL is the default. You must specify POINTER=INDX

or SYMB for any LCHILD statement following an index target segment type; no default is provided for this part of the index relationship. When the POINTER= keyword is omitted from an LCHILD statement which establishes a unidirectional or physically paired bidirectional logical relationship, POINTER=NONE is the default. When the POINTER= keyword is omitted or specified as NONE for an LCHILD statement which establishes a virtually paired bidirectional logical relationship, POINTER=SNGL is the default.

Restrictions:

- For PHDAM and PHIDAM databases, only the operands INDX and NONE are supported. All other operands are treated as if errors are present.
- For DEDB secondary index databases, only the SYMB operand is supported.

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

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

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

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

Requirement: If the target database is a HALDB, the index database must be defined as a HALDB index by use of the PSINDEX parameter in the DBD statement ACCESS parameter.

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

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

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

Restriction: This parameter is not allowed for virtual pairing when using PHDAM and PHIDAM databases, because they only support physical pairing.

INDEX=
Is specified on LCHILD statements for an Index DBD generation only. The fldname parameter specifies the name of the sequence field of a HIDAM root segment type during DBD generation of the primary index for a HIDAM database, or the name of an indexed field, defined through an XDFLD statement in an index target segment type during DBD generation of a secondary index database. This parameter is not needed for a primary index of a PHIDAM database.

RKSIZE=
Specifies the root key size of the target database. This parameter is required for partitioned secondary index (PSINDEX) databases only, and is invalid for any other database type.

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

Restriction: This parameter is not allowed for virtual pairing when using PHDAM and PHIDAM databases, because they only support physical pairing.

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

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

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

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

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

MULTISEG=
Identifies a set of LCHILD and XDFLD statements belonging to a multiple secondary index segment group. Valid values for the MULTISEG= parameter are YES or NO. NO is the default.

YES
Identifies the LCHILD/XDFLD statement pair as a member of a multiple secondary index segment group.

NO
Identifies the LCHILD/XDFLD statement pair not belonging to a multiple secondary index segment group.

Restriction: The MULTISEG= parameter is valid only on a LCHILD statement for a DEDB database. If MULTISEG= is specified for a database that is not a DEDB database, the DBDGEN utility terminates with an error message.

REMARKS=
Optional user comments. A 1- to 256-character field.

If your comments contain special characters, such as commas or blank spaces, enclose the full comment string in single quotation marks.

The value specified on the REMARKS keyword cannot contain the following characters:

- Single quotation marks, except when they are used to enclose the full comment string. If a single quotation mark is entered before the end of the full comment string, the remainder of the comment string is truncated. The following examples show correct and incorrect usages of single quotation marks on the REMARKS keyword:

  CORRECT
  ```
  REMARKS='These remarks apply to the XYZ application'
  ```
  INCORRECT
  ```
  REMARKS='These remarks apply to the 'XYZ' application'
  ```
- Double quotation marks.
- Less than (<) symbols.
- Greater than (>) symbols.
- Ampersands (&).

FIELD statements

The FIELD statement defines a field within a segment type. Fields are referred to by PSBs when defining sensitivity to the fields or by an application program in a DL/I call segment search argument.

The FIELD statement must meet the following conditions:

- A maximum of 1,000 FIELD statements that include the **NAME** parameter and XDFLD statements combined can be defined for all segments in a DBD generation.
- A maximum of 20,000 XDFLD statements and all FIELD statements combined can be defined for all segments in a DBD generation.
- A maximum of 255 FIELD statements and XDFLD statements combined can be defined for any segment type.

A unique sequence field must be defined for the root segment types of HISAM, HIDAM, PHIDAM, HIDAM Primary INDEX, SHISAM, DEDB, and non-terminal-related MSDB databases. Root segment types in an HDAM database do not need a key field defined; if a key field is defined, it does not have to be unique.

The use of /SX to define unique secondary indexes in HDAM, HIDAM, PHDAM, and PHIDAM databases causes a 4-byte RBA of the index source segment to be included as part of the key of the index record. The use of /CK to define unique secondary indexes in HISAM, HDAM, HIDAM, PHDAM, and PHIDAM databases does the same. In a PSINDEX, the /SX specification causes an 8-byte ILK to be used instead of a 4-byte RBA.

PSINDEX entries also contain the root key of the target segment.

FIELD statements are used in DBD generation:

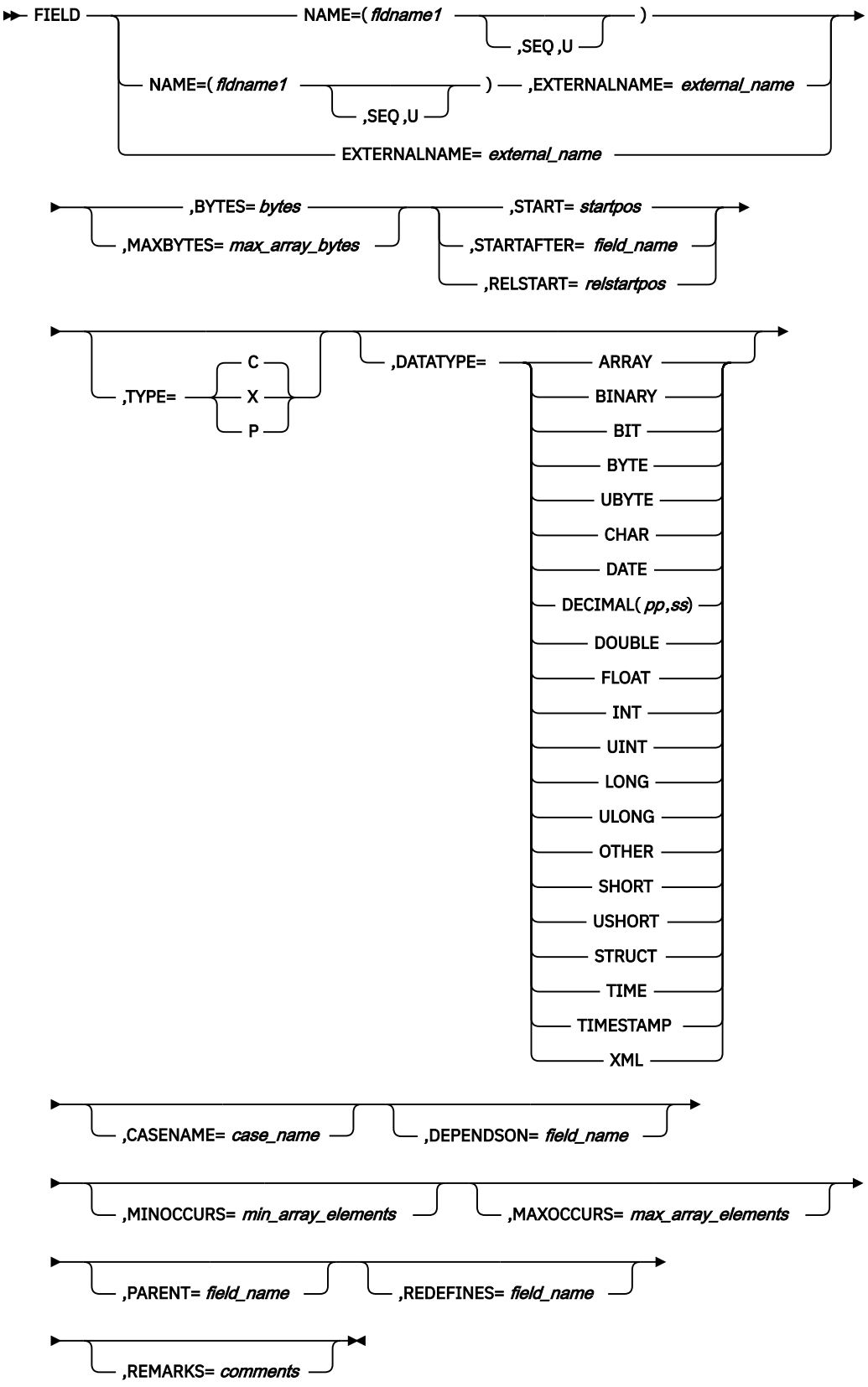
- To define fields of a segment type as that segment type is seen when it is accessed from its physical parent segment.

- To define the fields of a real logical child segment type in a virtually paired logical relationship as seen when that segment type is accessed from its logical parent. The FIELD statements must immediately follow the SEGM statement defining the virtual logical child.
- To define system-related fields that are used for secondary indexing.

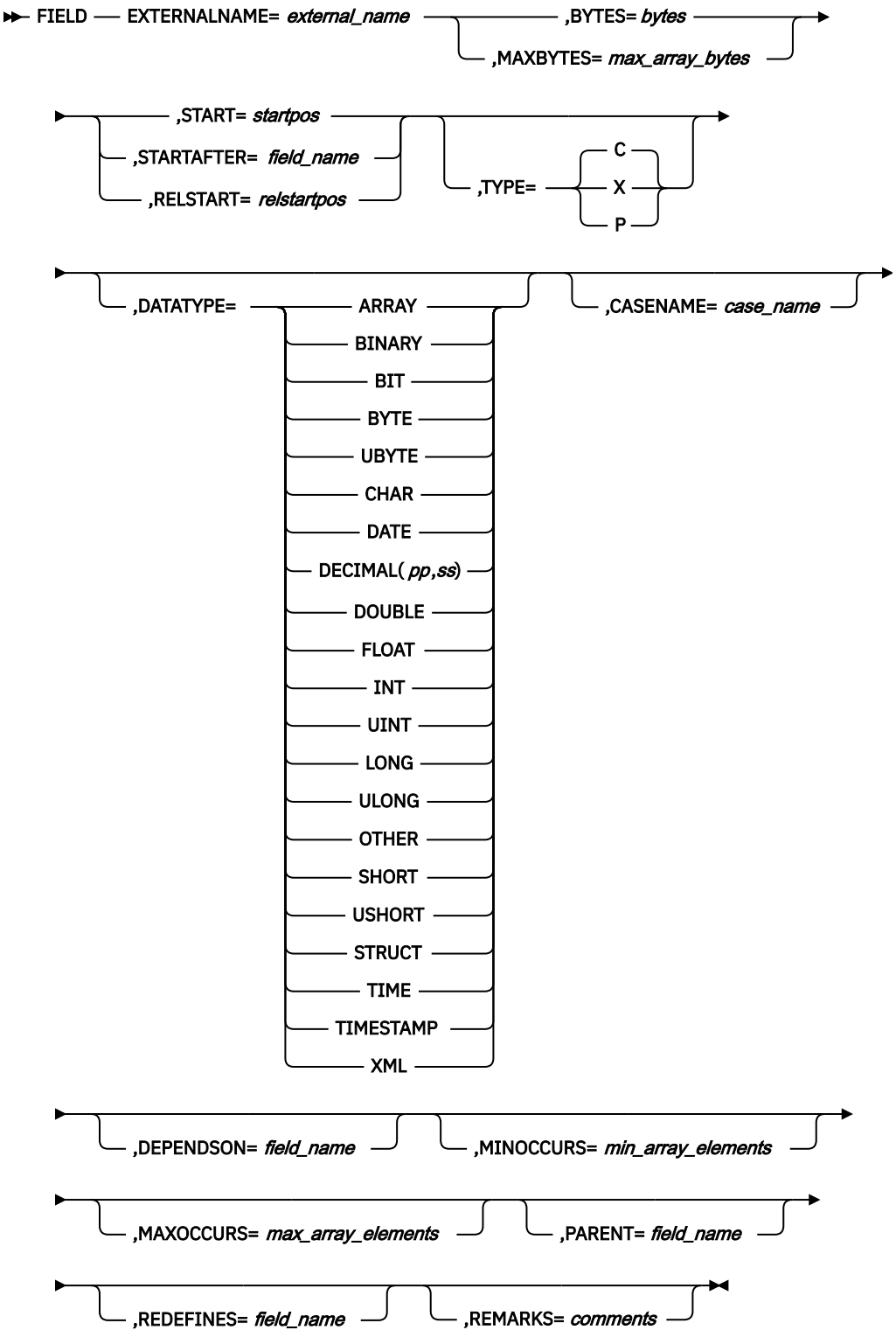
The “FIELD statement parameter descriptions” on page 113 are documented following the syntax diagrams below.

The format of the FIELD statement for each database type is shown in the following syntax diagrams.

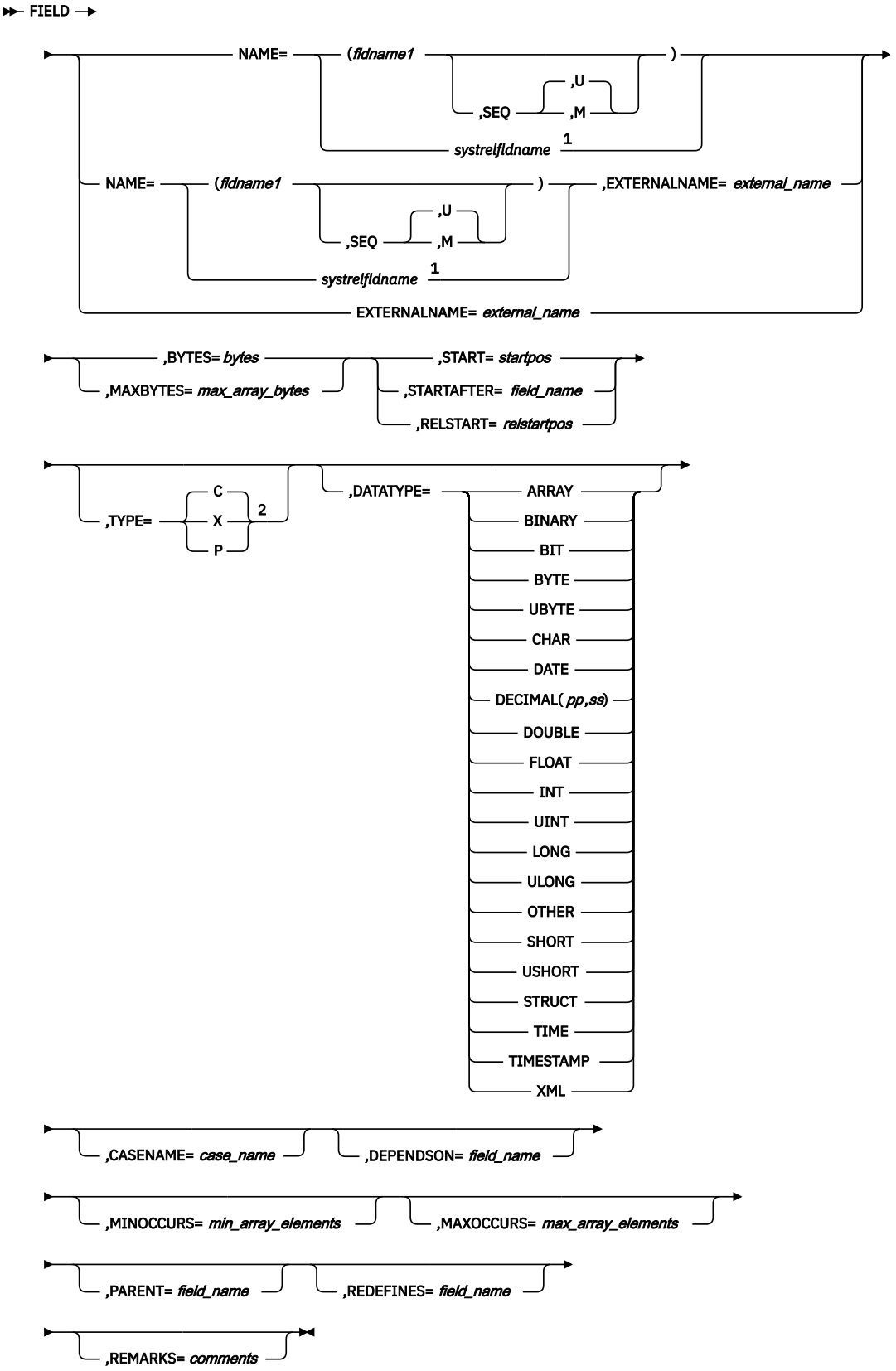
DEDB database FIELD statement



GSAM database FIELD statement



HDAM and PHDAM database FIELD statement

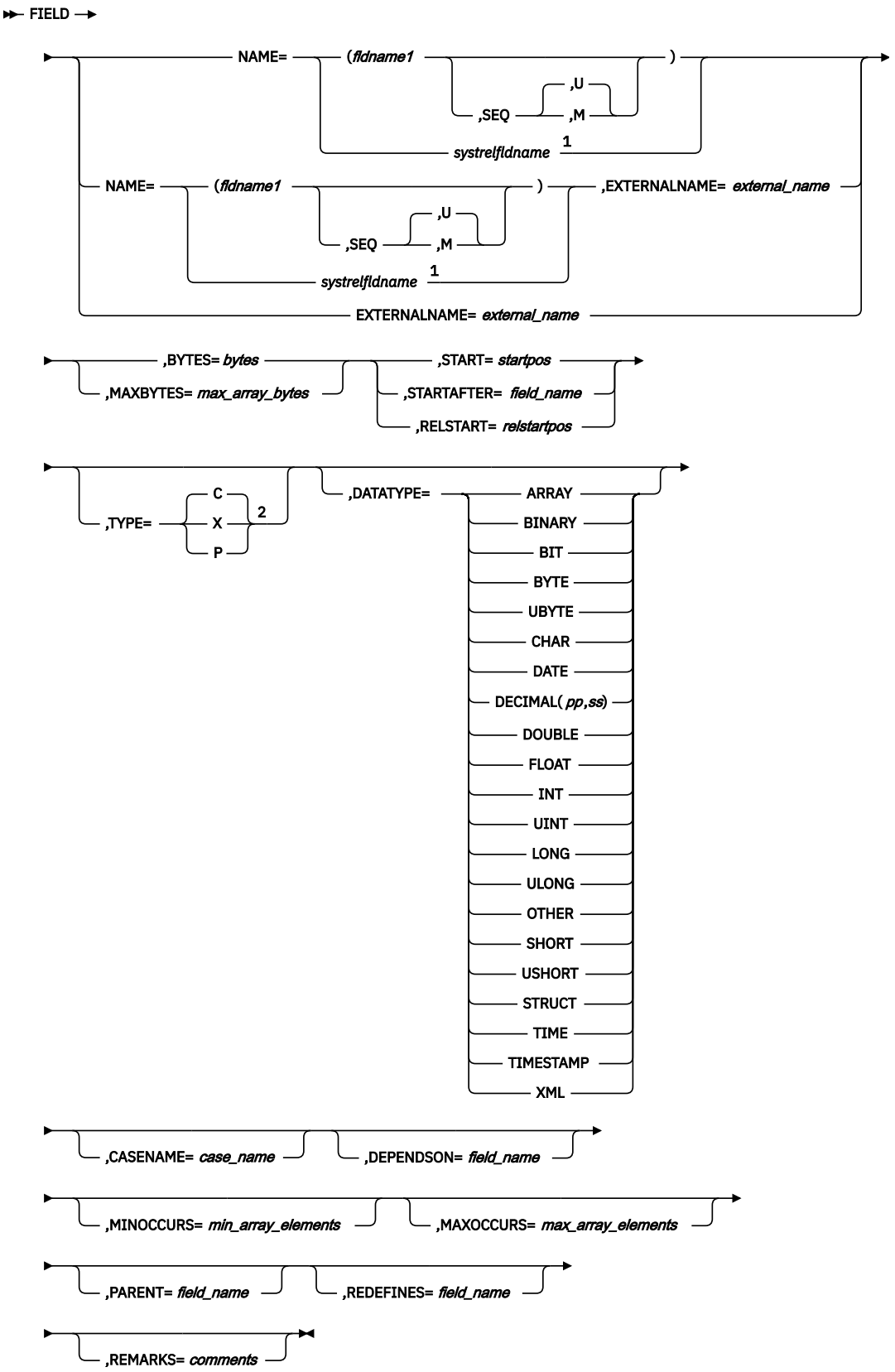


Notes:

[1] A system related field used for secondary indexing.

[2] The TYPE=parameter is ignored for fields with a *systrelfldname*.

HIDAM and PHIDAM database FIELD statements

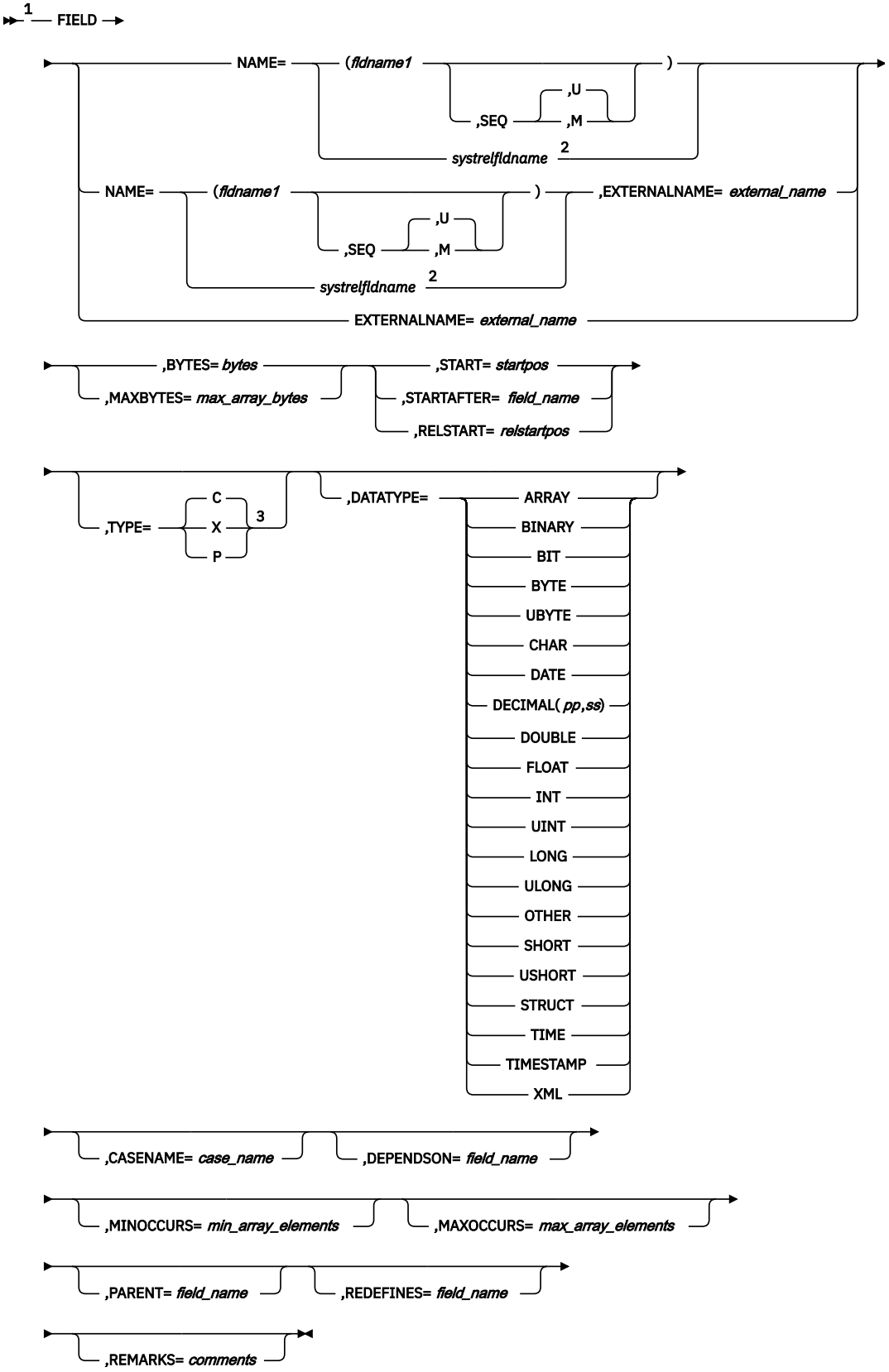


Notes:

[1] A system related field used for secondary indexing.

[2] The TYPE=parameter is ignored for fields with a *systrelfldname*.

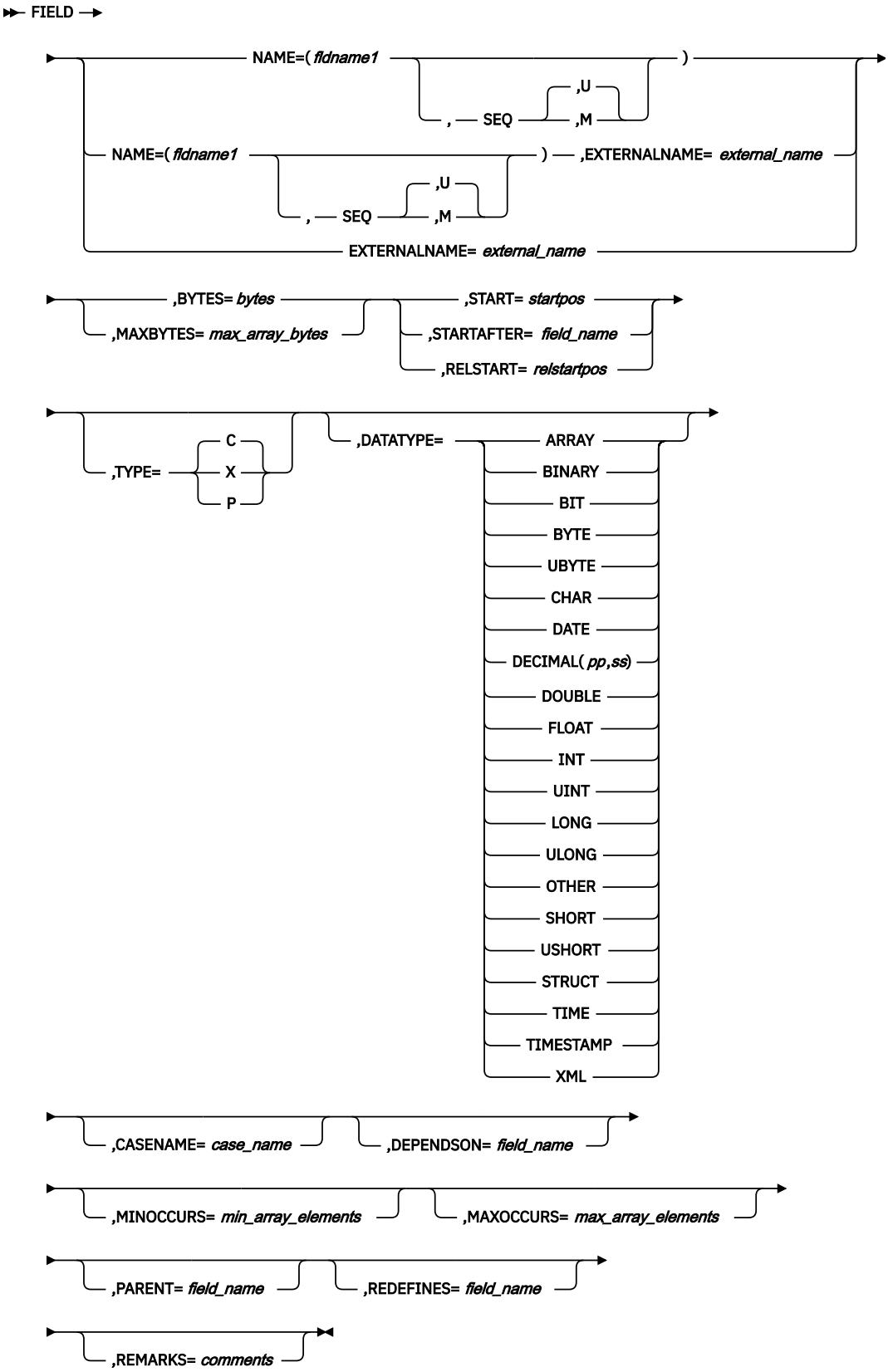
HISAM database FIELD statement



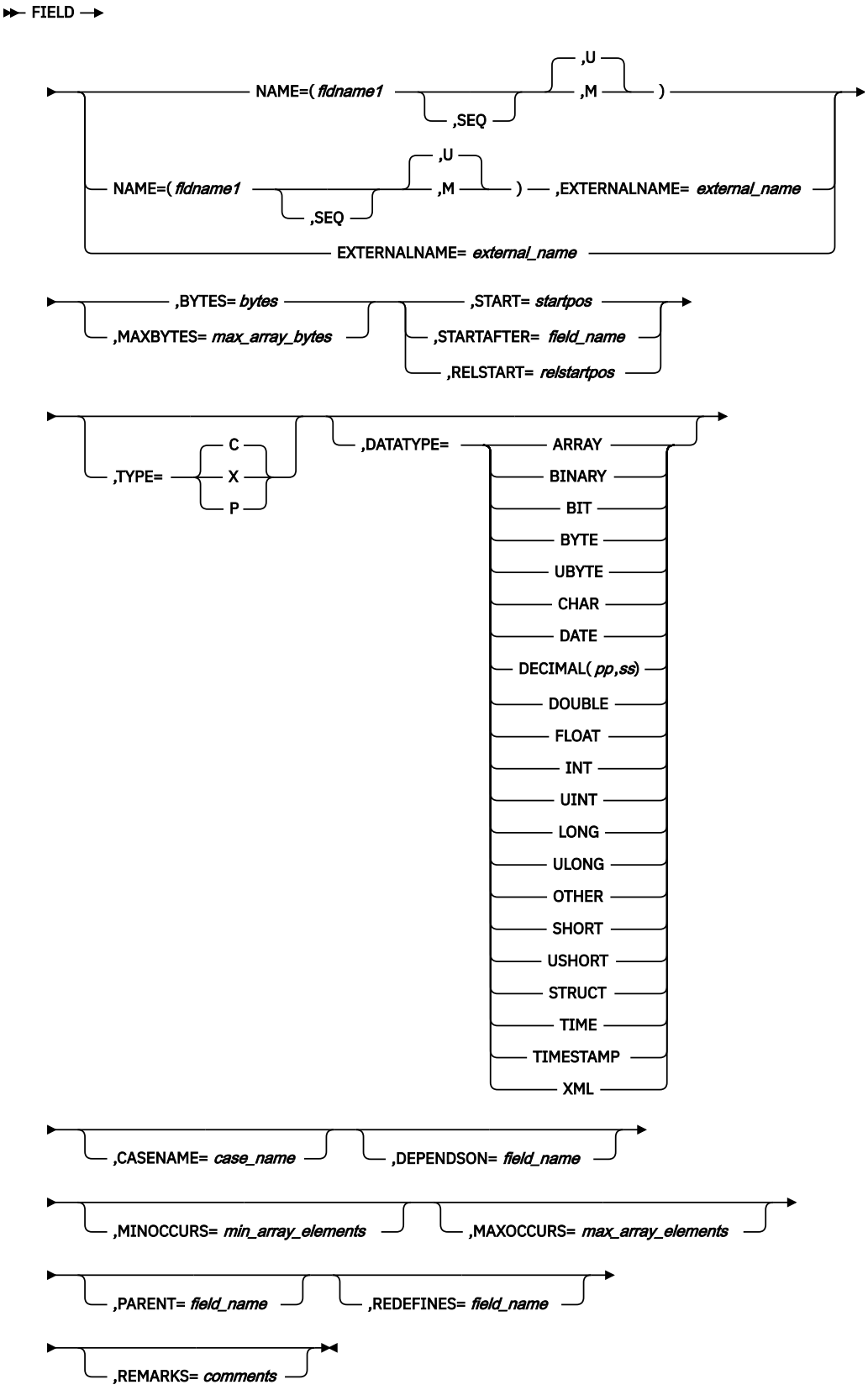
Notes:

[1] Only CK can be coded for the *systrelfldname* field.
[2] A system related field used for secondary indexing.
[3] The TYPE=parameter is ignored for fields with a *systrelfldname*.

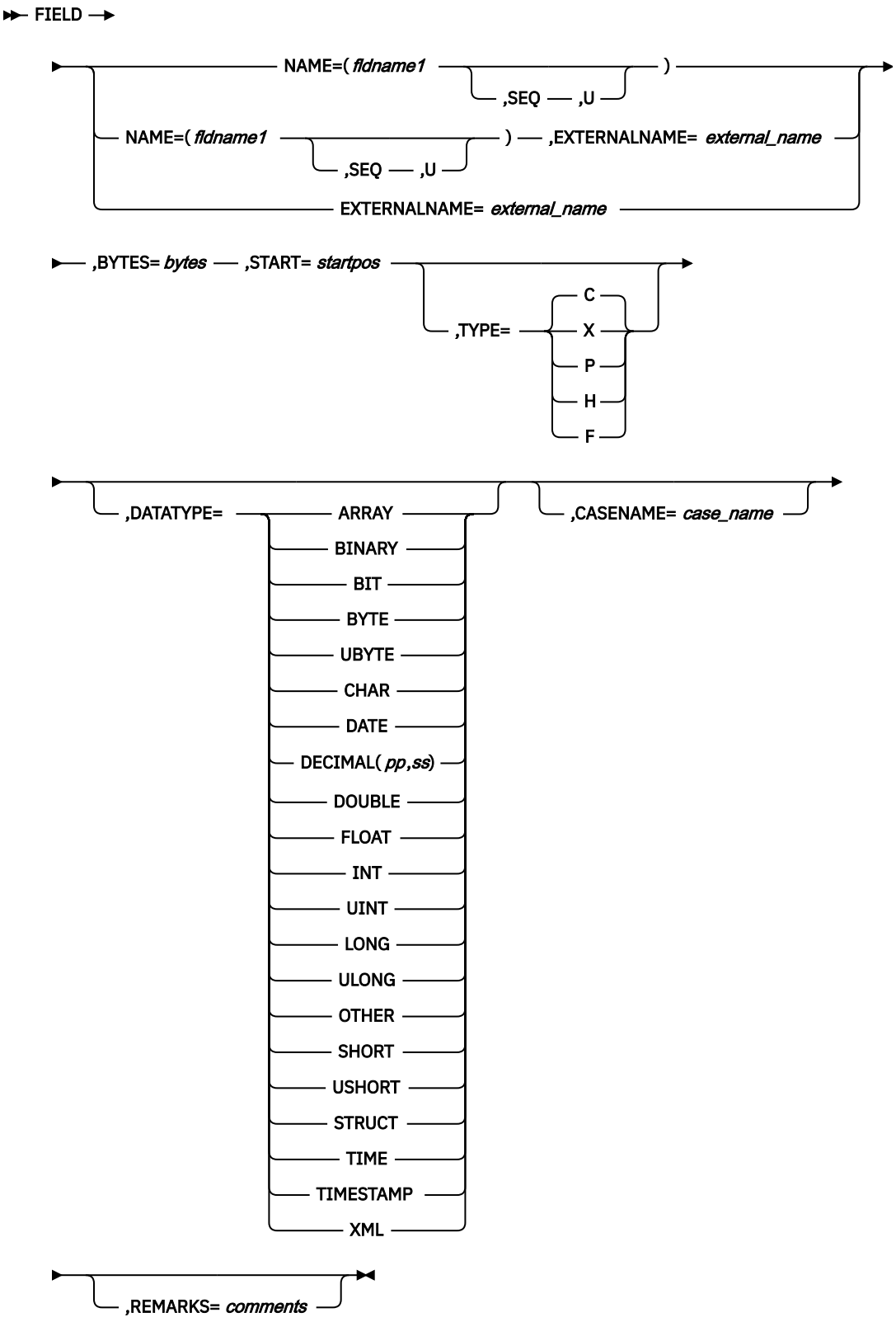
HSAM/SHSAM database FIELD statement



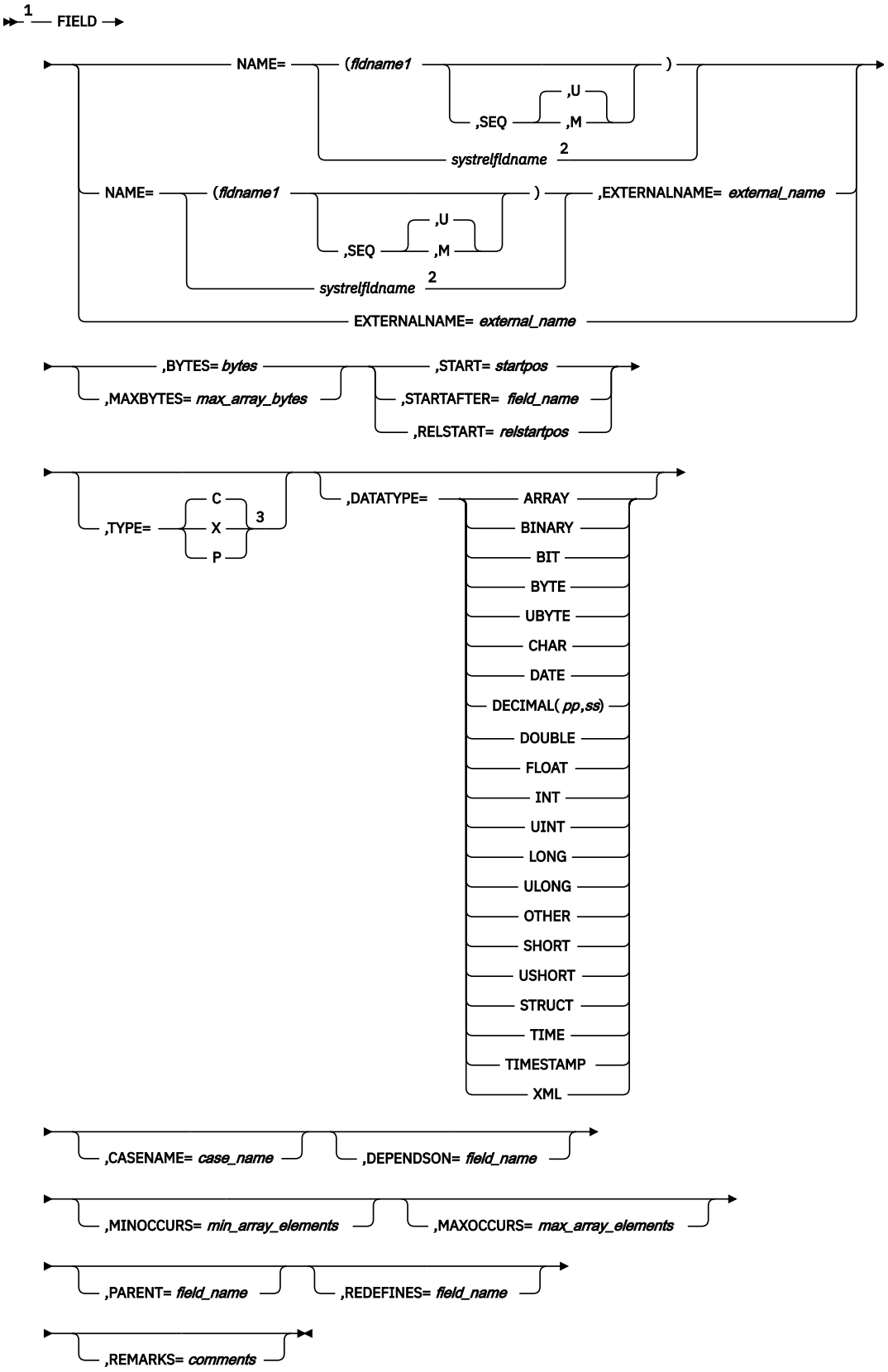
INDEX/PSINDEX database FIELD statement



MSDB database FIELD statement



SHISAM database FIELD statement



Notes:

[1] Only CK can be coded for the *systrelfldname* field.
[2] A system related field used for secondary indexing.
[3] The TYPE=parameter is ignored for fields with a *systrelfldname*.

FIELD statement parameter descriptions

BYTES=
Specifies the length of the field being defined in bytes. For fields other than system-related fields, BYTES must be a valid self-defining term whose value does not exceed 255.

If a concatenated key or a portion of a concatenated key of an index source segment type is defined as a system-related field, the value specified can be greater than 255, but must not exceed the length of the concatenated key of the index source segment.

A case in which the byte length can be greater than 255 is when the column is defined as not searchable by IMS. These columns cannot be defined as primary keys and cannot have the NAME keyword specified.

The length of a /SX system-related field is always 4 bytes; therefore, when specified, the BYTES parameter is disregarded.

If this field is defined as either a structure or an array by STRUCT or ARRAY, the value specified on BYTES must be greater than or equal to the sum total of the bytes of all fields contained in the structure or array.

When XML, the BYTES parameter is optional and the valid values for BYTES range from 0 to the maximum size of the segment. If the BYTES parameter is omitted when XML, BYTES and MAXBYTES are not allowed.

CASENAME=
The name of the map case that this field belongs to when alternative mappings are defined for the fields in a segment. CASENAME is valid and required only to associate a FIELD statement with the preceding DFSCASE statement that defines the map case to which this field belongs. The value of CASENAME must match the value specified on the NAME parameter of the DFSCASE statement.

DATATYPE=
An optional 3- to 9-character alphanumeric field that specifies the external data type of the field.

If DECIMAL is specified on the DATATYPE parameter, the default INTERNALTYPECONVERTER is signed PACKEDDECIMAL.

If DATE, TIME, or TIMESTAMP is specified on the DATATYPE parameter, you must specify either LONG or CHAR on the INTERNALTYPECONVERTER parameter in the DFSMARSH statement or specify a USERTYPECONVERTER. If a DFSMARSH statement is not included for this field, INTERNALTYPECONVERTER=LONG is the default. When LONG is used, the value is stored on DASD as the number of milliseconds since January 1, 1970.

If XML is specified on the DATATYPE parameter, the default INTERNALTYPECONVERTER is XML_CLOB, which is the only valid value when DATATYPE=XML is specified.

If STRUCT or ARRAY is specified on the DATATYPE parameter, the default INTERNALTYPECONVERTER is STRUCT or ARRAY, respectively, which are the only valid values when either one is specified on the DATATYPE parameter.

For all other values for DATATYPE, the value is used as the default INTERNALTYPECONVERTER.

If TYPE=C, DATATYPE defaults to CHAR. For any other specification of the TYPE parameter, DATATYPE defaults to BINARY.

If the DFSMARSH statement specifies USERTYPECONVERTER, the BYTES value is not validated for the data type of BIT, BYTE, UBYTE, DOUBLE, FLOAT, INT, UINT, LONG, ULONG, and SHORT.

Valid values are:

ARRAY
When ARRAY is specified:

- The NAME parameter is not supported
- The EXTERNALNAME parameter is required

- The byte value specified on either the BYTES or MAXBYTES parameter must be equal to or greater than the sum total of the bytes of all fields contained in the array.

The MSDB database type does not support the ARRAY data type.

BINARY
If TYPE=P or TYPE=X is specified, BINARY is the default value of the DATATYPE parameter.

BIT

If you specify BIT, you must also specify BYTES=1.

If you specify BIT, you must also specify BYTES=1 unless the DFSMARSH statement specifies USERTYPECONVERTER.

BYTE

If you specify BYTE, you must also specify BYTES=1.

If you specify BYTE, you must also specify BYTES=1 unless the DFSMARSH statement specifies USERTYPECONVERTER.

UBYTE

If you specify UBYTE, you must also specify BYTES=1.

If you specify UBYTE, you must also specify BYTES=1 unless the DFSMARSH statement specifies USERTYPECONVERTER.

CHAR
If TYPE=C is specified, CHAR is the default value of the DATATYPE parameter.

DATE
When DATE is specified, you must also specify BYTES=8, unless you also specify a DFSMARSH statement that includes either INTERNALTYPECONVERTER=CHAR or USERTYPECONVERTER=*convertername*.

DECIMAL(*pp*,*ss*)

pp
Precision. A 1- to 2-byte numeric field greater than 0.

ss
Scale. A 1- to 2-byte numeric field greater than or equal to 0. The value specified for *ss* cannot be greater than the value of *pp*.

You must specify a value on the BYTES parameter that matches the decimal format that is used.

The default decimal format is signed packed decimal. To calculate the required value of the BYTES parameter for the signed packed decimal format, use the following formula: *length* = ceiling ((*pp* + 1) / 2)

The default decimal format can be changed by specifying the INTERNALTYPECONVERTER parameter.

When the zoned decimal format is used, as specified by INTERNALTYPECONVERTER=ZONEDDECIMAL, use the following formula to calculate the value of the BYTES parameter: *length* = *pp*

DOUBLE

If you specify DOUBLE, you must also specify BYTES=8.

If you specify DOUBLE, you must also specify BYTES=8 unless the DFSMARSH statement specifies USERTYPECONVERTER.

FLOAT

If you specify FLOAT, you must also specify BYTES=4.

If you specify FLOAT, you must also specify BYTES=4 unless the DFSMARSH statement specifies USERTYPECONVERTER.

INT

If you specify INT, you must also specify BYTES=4.

If you specify INT, you must also specify BYTES=4 unless the DFSMARSH statement specifies USERTYPECONVERTER.

UINT

If you specify UINT, you must also specify BYTES=4.

If you specify UINT, you must also specify BYTES=4 unless the DFSMARSH statement specifies USERTYPECONVERTER.

LONG

If you specify LONG, you must also specify BYTES=8.

If you specify LONG, you must also specify BYTES=8 unless the DFSMARSH statement specifies USERTYPECONVERTER.

ULONG

If you specify ULONG, you must also specify BYTES=8.

If you specify ULONG, you must also specify BYTES=8 unless the DFSMARSH statement specifies USERTYPECONVERTER.

OTHER
Specifies the use of a user-defined data type. When OTHER is specified, a DFSMARSH statement must also be specified with a user-provided type converter specified on the USERTYPECONVERTER parameter.

SHORT

If you specify SHORT, you must also specify BYTES=2.

If you specify SHORT, you must also specify BYTES=2 unless the DFSMARSH statement specifies USERTYPECONVERTER.

USHORT
If you specify USHORT, you must also specify BYTES=2.

STRUCT
When STRUCT is specified, you cannot also specify the SEQ parameter if this structure field contains a dynamic array field as a child. Dynamic array fields are defined with DATATYPE=ARRAY and the DEPENDSON and MAXBYTES parameters, among others.

Also, the byte value specified on either the BYTES or MAXBYTES parameter must be equal to or greater than the sum total of the bytes of all fields contained in the structure.

The MSDB database type does not support the STRUCT data type.

TIME
When TIME is specified, you must also specify BYTES=8, unless you also specify a DFSMARSH statement that includes either INTERNALTYPECONVERTER=CHAR or USERTYPECONVERTER=*convertername*.

TIMESTAMP
When TIMESTAMP is specified, you must also specify BYTES=8, unless you also specify a DFSMARSH statement that includes either INTERNALTYPECONVERTER=CHAR or USERTYPECONVERTER=*convertername*.

XML

Restriction: DATATYPE=XML is not supported when the NAME parameter is specified.

DEPENDSON
Specifies the name of a field that defines the number of elements in a dynamic array. The FIELD statement of the referenced field must precede the FIELD statement that specifies the DEPENDSON parameter. The name specified must be the value, whether explicitly defined or accepted by default, of the EXTERNALNAME parameter in the definition of the referenced field.

The DEPENDSON parameter is valid only when ARRAY is also specified. DEPENDSON is required if the values of MINOCCURS and MAXOCCURS are different.

The field referenced by the DEPENDSON parameter must be defined with one of the following DATATYPE values:

- INT
- SHORT
- LONG
- UINT
- USHORT
- ULONG
- DECIMAL with either (pp) or (pp,ss) specified, where ss is either 0 or 00.

The MSDB database type does not support the DEPENDSON parameter.

EXTERNALNAME=
An optional alias for the NAME= parameter. Java application programs use the external name to refer to the field. The external name is stored only in the IMS catalog, not in the database that you are defining.

The EXTERNALNAME parameter is required only when either the NAME parameter is not specified or the field is defined in a GSAM database. If the NAME parameter is not specified, you cannot search for this field.

Specify an external name as a 1- to 128-character uppercase alphanumeric string. An external name can include underscore characters.

External names must be unique within a segment.

The default value of the EXTERNALNAME parameter is the value of the NAME parameter.

Restriction: External names cannot be reserved SQL keywords or begin with DFS.

If EXTERNALNAME is not specified and a reserved SQL keyword is specified in the NAME parameter, EXTERNALNAME accepts the NAME value as the default external name after appending "_COL" to the NAME value.

For a list of reserved SQL keywords that are restricted by the IMS Universal drivers, see Portable SQL keywords restricted by the IMS Universal JDBC drivers (Application Programming).

M or U
See the entry for **`U or M`** later in this topic.

MINOCCURS=
For ARRAY only, a required numeric value that specifies the minimum number of elements in an ARRAY. MINOCCURS must be lesser than or equal to MAXOCCURS.

MAXOCCURS=
For ARRAY only, a required numeric value that specifies the maximum number of elements in an ARRAY. MAXOCCURS must be greater than or equal to MINOCCURS and not zero.

MAXBYTES=
Specifies the maximum size of a field in bytes when the byte-length of the field instance can vary based on the number of elements in a dynamic array. MAXBYTES and BYTES are mutually exclusive.

The value of MAXBYTES must be greater than or equal to the maximum possible sum total of the byte values of all fields nested under this field.

The MAXBYTES parameter is required and valid only in the following cases:

- The field is defined as a dynamic array. A field is a dynamic array when the number of elements in the array can vary from one instance of the field to another. In the definition of a dynamic array, the **DEPENDSON** parameter references another field in the segment definition that will define the number of array elements for a given instance of the dynamic array.
- For a field defined as a static array or a structure that contains a nested field that is defined as a dynamic array.

The MSDB database type does not support the **MAXBYTES** parameter.

NAME=*fldname1*
Specifies the name of this field within a segment type. The name specified can be referred to by an application program in a DL/I call SSA. Field names must be unique within a segment definition. The *fldname1* value must be a 1- to 8-character alphanumeric value.

The **NAME** parameter is required on the following types of fields:

- Key-sequenced field types, which specify the **SEQ** parameter
- Field types that are referenced by a segment search argument (SSA)
- Field types that are referenced by in a SENFLD statement in a PSB
- Field types that are referenced by an XDFLD statement

For other field types, you can omit the **NAME** parameter when the **EXTERNALNAME** parameter is specified. Omitting the **NAME** parameter can save storage in the data management block (DMB) of a database. However, to be able to search on this field, you must specify the **NAME** parameter.

The **NAME** parameter cannot be specified on the following types of fields:

- Fields that are defined in a GSAM database. Use the **EXTERNALNAME** parameter instead.
- Fields that are defined as arrays. A field that is defined as an array includes ARRAY in the field definition.
- Fields that are defined as array elements. A field that is an array element specifies the name of an array field on the **PARENT** parameter in the FIELD statement.
- Fields that are defined as structures that contain one or more nested dynamic arrays. A field that is defined as a structure includes DATATYPE=STRUCT in the field definition.
- Fields that are contained in a structure that also contains a dynamic array. A field that is contained within a structure specifies the name of the structure field on the **PARENT** parameter in the FIELD statement.
- Fields that follow a dynamic array in a segment. A field that follows a dynamic array specifies the **STARTAFTER** parameter.
- Fields that include the **RELSTART** parameter to specify a starting position that is relative to the starting position of another field.
- Fields defined with XML.

PARENT=
Specifies the name of a field that is defined as a structure or array in which this field is contained. The referenced field must be defined with either DATATYPE=ARRAY or DATATYPE=STRUCT.

REDEFINES=

The name of the redefined field, as specified on the **EXTERNALNAME** parameter of the FIELD statement that defines the redefined field. The value can be specified as a 1- to 128-character alphanumeric string.

If the redefined field does not specify the **EXTERNALNAME** parameter, the value of the **NAME** parameter can be used. If the redefined field specifies both the **NAME** and **EXTERNALNAME** parameters with different values on each, the value of the **EXTERNALNAME** parameter must be used.

In the DBD generation input order, the FIELD statement of the field that is being redefined must precede the FIELD statement that specifies the **REDEFINES** parameter.

This field must be the same length as the field that is being redefined, as specified on the **BYTES** parameter in each FIELD statement.

You cannot redefine a field that has been defined as an ARRAY or that contains an ARRAY.

The MSDB database type does not support the **REDEFINES** parameter.

RELSTART=
Specifies the starting position of a field that is defined as an element of an array or, in some circumstances, a structure. Valid values are from 1 to 32767.

The value specified on RELSTART is the starting byte offset of the field relative to the start of the array or structure. For example, the first field in an array would typically specify RELSTART 1, even if the array that contains the field starts at byte 50 of a segment.

For fields that specify an array field as a parent, RELSTART is required.

For fields that specify a structure as a parent, RELSTART is required if the structure field is defined with RELSTART or STARTAFTER.

In the following example, the field DYNARRAY is a dynamic array. The field STRUCT01 is a structure. The fields FLD03 and FLD04 both specify STRUCT01 as a parent. Because a dynamic array precedes STRUCT01 in the segment, the starting offsets of FLD03 and FLD04 can be specified only relative to the start of STRUCT01.

```
FIELD EXTERNALNAME=ARRAYNUM,DATATYPE=DECIMAL(7,0),START=1,BYTES=4
FIELD EXTERNALNAME=DYNARRAY,DATATYPE=ARRAY,START=5,MAXBYTES=100
      MINOCCURS=10,MAXOCCURS=50,DEPENDSON=ARRAYNUM
FIELD EXTERNALNAME=FLD01,RELSTART=1,BYTES=2,PARENT=DYNARRAY
FIELD EXTERNALNAME=FLD02,STARTAFTER=DYNARRAY,BYTES=10
FIELD EXTERNALNAME=STRUCT01,DATATYPE=STRUCT,STARTAFTER=FLD02,BYTES=10
FIELD EXTERNALNAME=FLD03,RELSTART=1,BYTES=5,PARENT=STRUCT01
FIELD EXTERNALNAME=FLD04,RELSTART=6,BYTES=5,PARENT=STRUCT01
```

START, STARTAFTER, and RELSTART are mutually exclusive.

REMARKS=
Optional user comments. A 1- to 256-character field.

If your comments contain special characters, such as commas or blank spaces, enclose the full comment string in single quotation marks.

The value specified on the REMARKS keyword cannot contain the following characters:

- Single quotation marks, except when they are used to enclose the full comment string. If a single quotation mark is entered before the end of the full comment string, the remainder of the comment string is truncated. The following examples show correct and incorrect usages of single quotation marks on the REMARKS keyword:

  CORRECT
  ```
  REMARKS='These remarks apply to the XYZ application'
  ```

  INCORRECT
  ```
  REMARKS='These remarks apply to the 'XYZ' application'
  ```
- Double quotation marks.
- Less than (<) symbols.
- Greater than (>) symbols.
- Ampersands (&).

SEQ
A subparameter of NAME, SEQ identifies this field as a sequence field in the segment type. FIELD statements containing the keyword SEQ must be the first FIELD statements following a SEGM statement in a DBD generation input deck.

If the sequence field of a real logical child segment consists of any part of the concatenated key of the logical parent, you must specify the PHYSICAL parameter in the SEGM statement in order for the logical child to include the concatenated key of the logical parent with the logical child in storage.

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

A sequence field must be specified for a virtual logical child segment type if, when accessing a logical child segment from its logical parent, one requires real logical child segments to be retrieved in an order determined by data in a field or fields of the real logical child segments. This sequence field can include any part of the segment as it appears when viewed from the logical parent (that is, the concatenated key of the physical parent of the real logical child followed by any intersection data). Because it might be necessary to describe the sequence field of a logical child segment as accessed from its logical parent segment in noncontiguous pieces, multiple FIELD statements with the SEQ parameter present are permitted. Each statement must contain a unique *fldname1* parameter.

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

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

In a DEDB, SEQ must be used in the root segment and can be specified in any direct dependent segment.

Restrictions:

- SEQ cannot be specified for the sequential dependent segment
- SEQ cannot be specified for a field that is defined as a structure that contains a field that is defined as a dynamic array. Structure fields are defined by DATATYPE=STRUCT. Dynamic array fields are defined by DATATYPE=ARRAY and the DEPENDSON and MAXBYTES parameters, among others.

START=

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

START can be used for a system-related field, to describe a portion of the concatenated key as a field in an index source segment type. If used in this way, START specifies the starting position of the relevant portion of the concatenated key relative to the beginning of the concatenated key. The first byte of the concatenated key is considered to have a position of one. It must be a numeric term whose value does not exceed the length of the concatenated key plus one. Subtract the value specified in the **BYTES** parameter. The starting position parameter for the /SX system-related field is disregarded.

START, STARTAFTER, and RELSTART are mutually exclusive.

When XML, the START parameter is optional and START 0 can be specified. If the START parameter is omitted when XML, START 0 is the default.

STARTAFTER=

When the starting byte offset of a field cannot be calculated because the field starts after a dynamic array, specifies the name of the field that directly precedes this field in the segment. The name cannot be the name provided on the NAME keyword.

STARTAFTER is required and valid only when the starting position of a field cannot be calculated because the field is preceded at a prior offset by a field defined as a dynamic array.

Dynamic arrays make it impossible to calculate the starting offsets of subsequent fields in a segment, because the byte lengths of dynamic arrays can vary from one instance of a segment to another. The columns of dynamic array fields can be identified by the inclusion of the DEPENDSON and MAXBYTES parameters.

The STARTAFTER parameter cannot be specified on fields that define an array field as a parent. Instead, specify the RELSTART parameter.

START, STARTAFTER, and RELSTART are mutually exclusive.

The STARTAFTER data is not returned if the GUR application is using IMS Universal Drivers (UDB).

systrelfldname

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

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

  The following are sample concatenated keys for a given index source segment type:

  – Root key (10 bytes)
  – Dependent key (3 bytes)
  – Dependent key (3 bytes)
  – Dependent key (3 bytes)

  If three system-related fields were to consist of bytes 2 through 8 of the root key, byte 1 of the second key and bytes 2 and 3 of the fourth key, the FIELD statements specifying these fields could be as follows:

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

NAME=/CK2
BYTES=1
START=11

NAME=/CK3
BYTES=2
START=18
```

  You can then specify the three system-related fields defined for use in the subsequence or duplicate data fields of index pointer segments by including the names of the system-related fields in lists for the subsequence or duplicate data fields on an XDFLD statement.
- The second type of system-related field is defined within an index source segment type to ensure uniqueness of sequence field keys in a secondary index. The name specified for this type of system-related field must begin with the characters /SX, and the name specified can be up to eight characters in length. When this type of system-related field is defined in an index source segment type, IMS generates a unique 4-byte value, and places it in the subsequence field of the index pointer segment generated from an index source segment.

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

For Fast Path secondary indexing, only a /CK field is valid and the /SX field is not valid.

TYPE=

Determines the type of character that IMS uses to mask or pad the data in this field.

If the DATATYPE parameter is not explicitly set, the TYPE parameter also determines the default value of DATATYPE; however, TYPE does not otherwise affect how data is stored, converted, or presented to application programs.

For example, when application programs that use field-level sensitivity are not sensitive to this field, IMS can mask the data in a field with either X'00', X'40, or, for MSDBs, halfword or fullword binary data.

When an application program is sensitive to one or more fields in a segment, IMS masks fields if one of the following conditions is met:

- On an insert call, the segment contains fields that the application program is not sensitive to.
- On a call that replaces a variable-length segment with a segment that is longer than the existing segment, the increased portion of the segment contains fields that the application program is not sensitive to.
- On a call that retrieves a variable-length segment that does not contain the field.

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

All DL/I calls perform field comparisons on a byte-by-byte binary basis. No check is made by IMS to ensure that the data contained within a field is of the type specified by this parameter, except when the defined field is used with field sensitivity or is in an MSDB.

You can specify the following values on the TYPE parameter:

X

Specifies hexadecimal data. When X is specified, if IMS needs to fill unused bytes in the field, IMS right justifies the value and fills the unused bytes to the left of the value with X'00'. For example, a 3-byte value X'543210' in a 5-byte field is written out as X'0000543210'.

P

Packed decimal data. When P is specified, if IMS needs to fill unused bytes in the field, IMS right justifies the value and fills the unused bytes to the left of the value with X'00'. For example, a 3-byte value X'54321C' in a 5-byte field is written out as X'000054321C'.

C

Specifies alphanumeric data or a combination of types of data. When C is specified, if IMS needs to fill unused bytes in the field, IMS left justifies the value and fills the unused bytes to the right of the value with X'40'. For example, a 3-byte value X'F5F4F3' in a 5-byte field is written out as X'F5F4F34040'.

F

Specifies binary fullword data.

An arithmetic field cannot be overlapped by another field in a segment definition; that is, another field cannot be defined to start or end between the starting and ending byte offsets of a field that specifies TYPE=F.

H

Specifies binary halfword data.

An arithmetic field cannot be overlapped by another field in a segment definition; that is, another field cannot be defined to start or end between the starting and ending byte offsets of a field that specifies TYPE=H.

For MSDB databases, types X, C, P, H, and F are valid, with the following rules applying:

- Only a C or X field can contain another field.
- A single field can have multiple definitions as long as no more than one definition is arithmetic (types P, H, and F).
- If a field contains any part of an arithmetic field, it must contain the entire field.
- The sequence field must be TYPE=C or X.
- The sequence field cannot be part of any other field.
- SSA and FSA comparisons of arithmetic fields use arithmetic rather than logical compare operations.
- Initial loading and call processing routines test for valid digits and X and P type fields.
- The following rules apply to the MSDB field length:
  - TYPE=X: BYTES=1 to 256
  - TYPE=P: BYTES=1 to 16
  - TYPE=C: BYTES=1 to 256
  - TYPE=F: BYTES=4
  - TYPE=H: BYTES=2
  - Field types F and H must have explicit length specifications.
  - Fields should be aligned on appropriate boundaries for performance optimization if they are involved in compare or arithmetic operations and are a fullword or halfword long. The beginning of the segment is aligned on a fullword boundary.
- If the *systrelfldname* in the field statement is defined as either /SX or /CK, the TYPE= parameter is ignored and no type is set.

U or M

Subparameters of NAME, U and M qualify the type of sequence (SEQ) field that is being specified.

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

The parameter M indicates that duplicate values are allowed in the sequence field of occurrences of the segment type. For a root segment type, the sequence field of each occurrence must contain a unique value, except in HDAM. The root segment type in an HDAM database does not need a key field; if a key field is defined, it does not have to be unique.

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

Recommendation: Use unique sequence fields for all segments that participate in a logical relationship. This includes physical and logical parents as well as physical and logical child segments. Multiple sequence fields for a virtual logical child segment type must be uniformly defined as either unique or nonunique.

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

Related concepts

Defining DBD and PSB metadata to the generation utilities (Database Administration)

Related reference

"DFSMARSH statements" on page 127

The DFSMARSH statement defines the marshal attributes for a field.

XDFLD statements

Use the XDFLD statement only for secondary index relationships. The XDFLD statement defines the name of an indexed field that is associated to an index target segment type, identifies the index source segment type, and identifies the index source segment fields that are used in creating a secondary index.

In addition, information regarding suppressing the creation of index pointer segments is provided through this statement.

Restriction: This statement cannot be used to reference a segment in a DBD where ACCESS=INDEX, SHSAM, SHISAM, HSAM, or MSDB has been specified.

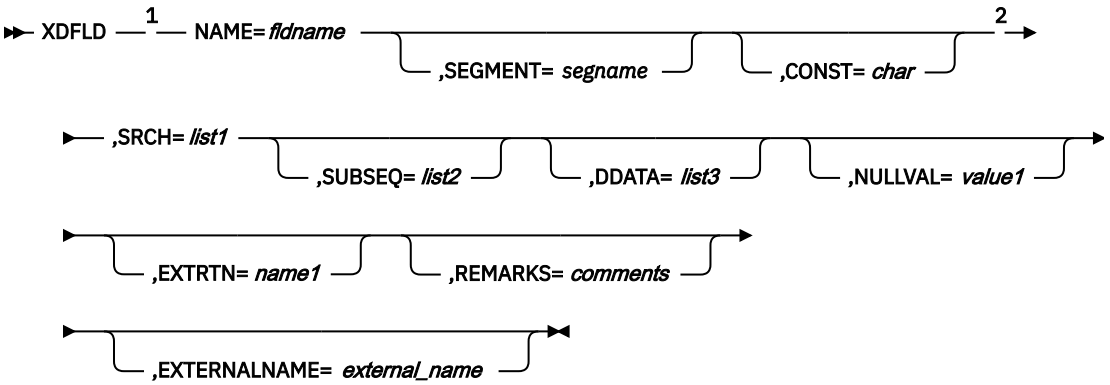
The XDFLD statement must meet the following conditions:

- A maximum of 32 XDFLD statements are allowed per SEGM statement.
- The number of XDFLD and all FIELD statements combined must not exceed 255 per SEGM statement, and must not exceed 20,000 per DBD.
- The number of FIELD statements that include the **NAME** parameter and XDFLD statements combined must not exceed 1,000 per DBD.

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

The format of the XDFLD statement is for each database type is shown in the following examples.

HISAM/SHISAM database XDFLD statement

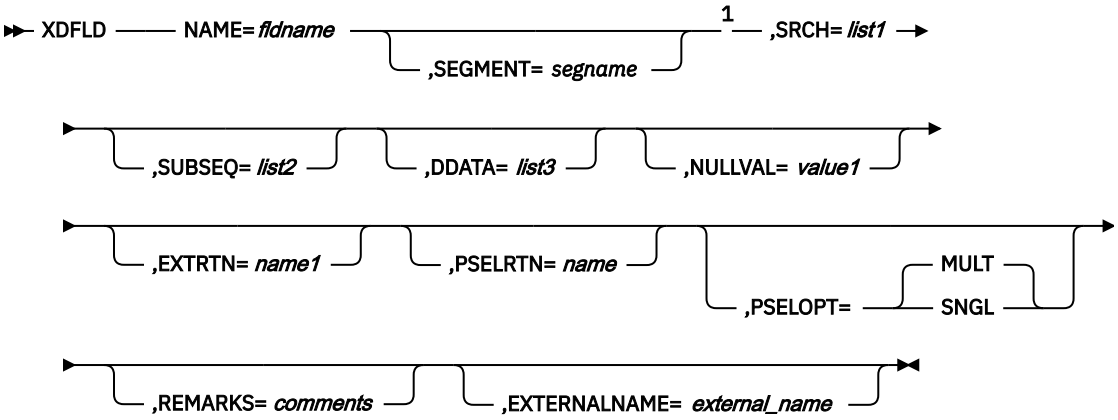


Notes:

[1] An XDFLD statement is not allowed during DBD generation of a simple HISAM database.

[2] The combined length of the CONSTANT, SEARCH, and SUBSEQUENCE fields must not exceed 240 bytes.

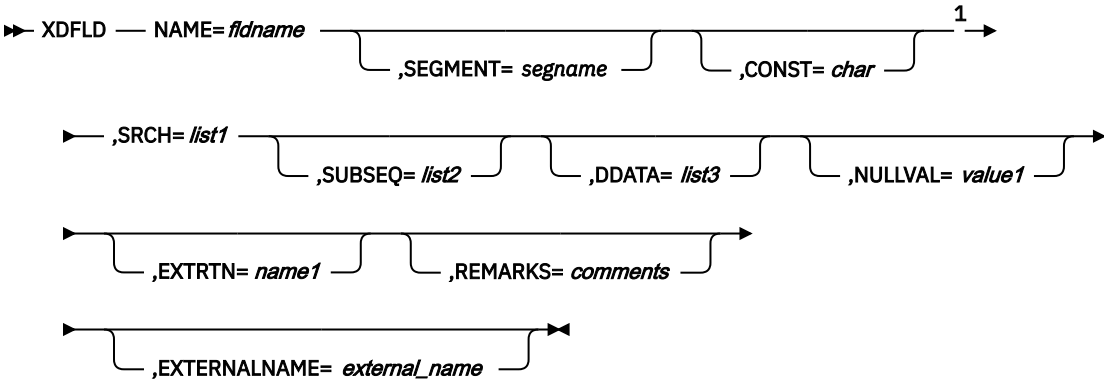
DEDB database XDFLD



Notes:

[1] The combined length of the SEARCH and SUBSEQUENCE fields must not exceed 240 bytes.

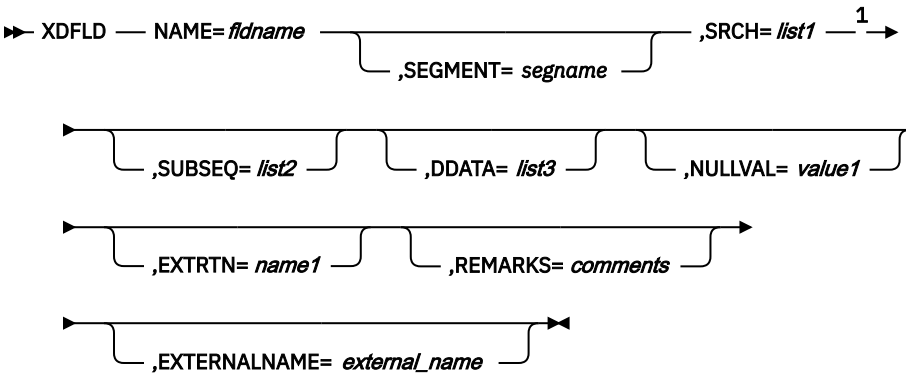
HDAM and HIDAM database XDFLD statement



Notes:

[1] The combined length of the CONSTANT, SEARCH, and SUBSEQUENCE fields must not exceed 240 bytes.

PHDAM and PHIDAM database XDFLD statement



Notes:

[1] The combined length of the SEARCH and SUBSEQUENCE fields must not exceed 240 bytes.

XDFLD statement parameter description

NAME=
Specifies the name of the indexed data field of an index target segment. The name specified actually represents the search field of an index pointer segment type as being a field in the index target segment type. You can use the name specified to qualify SSAs of calls for an index target segment type through the search field keys of index pointer segments. This enables accessing occurrences of an index target segment type through a primary or secondary processing sequence based on data contained in a secondary index. fldname must be a 1- to 8-character alphanumeric value.

Since the name specified is used to access occurrences of the index target segment type based on the content of a secondary index, the name specified must be unique among all field names specified for the index target segment type.

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

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

Restriction: CONST is not supported for HALDB or DEDB databases.

SRCH=
Specifies which field or fields of the index source segment you must use as the search field of a secondary index. *list1* must be a list of one to five field names defined in the index source segment type by FIELD statements. If two or more names are included, they must be separated by commas and enclosed in parentheses. The sequence of names in the list is the sequence in which the field values are concatenated in the index pointer segment search field. The sum of the lengths of the participating fields constitutes the index target segment indexed field length which must be reflected in segment search arguments.

For Fast Path secondary indexes, you can use part of a concatenated key as a value for the SRCH parameter by specifying the /CK operand, a starting byte, and a length in parentheses. For example, `SRCH=((/CK,1,3),(/CK,5,2),(/CK,9,2))` or `SRCH=((/CK,1,3),FLDNM)`.

SUBSEQ=
Specifies which, if any, fields of the index source segment you must use as the subsequence field of a secondary index. *list2* must be a list of one to five field names defined in the index source segment by FIELD statements. If two or more names are included, they must be separated by commas and enclosed in parentheses. The sequence of names in the list is the sequence in which field values are concatenated in the index pointer segment subsequence field. This parameter is optional.

For Fast Path secondary indexes, you can use part of a concatenated key as a value for the SUBSEQ parameter by specifying the /CK operand, a starting byte, and a length in parentheses. For example, `SUBSEQ=((/CK,1,3),(/CK,5,2),(/CK,9,2))` or `SUBSEQ=((/CK,1,3),FLDNM)`.

DDATA=
Specifies which, if any, fields of the index source segment you must use as the duplicate data field of a secondary index. *list3* must be a list of one to five field names defined in the index source segment by FIELD statements. If two or more names are included, they must be separated by commas and enclosed in parentheses. The sequence of names in the list is the sequence in which field values are concatenated in the index pointer segment duplicate data field. This parameter is optional.

NULLVAL=

Suppresses the creation of index pointer segments when the index source segment data used in the search field of an index pointer segment contains the specified value.

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

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

There is a slight difference in the case of packed fields. For packed fields, each field that composes the search field is considered to be a separate packed value.

Example: If the NULLVAL=X'9F' were specified in a case where the search field was composed of three 2-byte packed source fields, there would be no index entries with the search field value of X'999F999F999F' because all index entries containing a X'9F'would be suppressed.

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

The only form of the sign that is checked is the form specified.

Example: If X'9C' is specified, X'9F' does not cause suppression.

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

EXTRTN=

Specifies the name of a user-supplied index maintenance exit routine that is used to suppress the creation of selected index pointer segments. The parameter (name1) must be the name of a user-supplied routine which receives control whenever DL/I attempts to insert, delete or replace an index entry because of changes occurring in the indexed database. This exit routine can inspect the affected index source segment and decide whether an index pointer segment should be generated.

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

EXTERNALNAME=

Acts as an optional alias for the **NAME=** parameter. Java application programs use the external name to refer to the field. The external name is stored only in the IMS catalog, not in the database. The **EXTERNALNAME** parameter is optional. The **NAME** parameter must be specified.

Specify an external name as a 1- 26 character uppercase alphanumeric string. The following additional characters are also allowed:

- _ (underscore)
- $
- #
- @

External names must be unique within a segment.

The default value of the **EXTERNALNAME** parameter is the value of the **NAME** parameter.

Note: External names cannot be reserved SQL keywords or begin with DFS.

If **EXTERNALNAME** is not specified and a reserved SQL keyword is specified in the **NAME** parameter, **EXTERNALNAME** accepts the **NAME** value as the default external name after appending "_IDX" to the **NAME** value. For a list of reserved SQL keywords that are restricted by the IMS Universal drivers, see Portable SQL keywords restricted by the IMS Universal JDBC drivers.

PSELRTN=
Identifies the name of a user partition selection exit routine when user partitioning is requested for HISAM or SHISAM Fast Path secondary index databases.

PSELOPT=
Indicates how user partition databases in a user partition group are logically grouped for qualified GN calls with no SSA processing before the end of data is reached on the user partition databases. User partition databases are defined as part of a user partition group in the NAME= parameter on the LCHILD statement. This parameter applies to Fast Path secondary index databases only.

The PSELOPT= parameter can also be specified on the PCB statement with the PROCSEQD= parameter in a PSB. There is no default for the PSELOPT= parameter on the PCB statement with the PROCSEQD= parameter. If the PSELOPT= parameter is specified on both the XDFLD statement and the PCB statement with the PROCSEQD operand, the PSELOPT= parameter on the PCB statement takes precedence.

MULT
Indicates the selected user partition and its subsequent user partition databases in a user data partition group as they are physically defined in the NAME= parameter on the LCHILD statement of the primary DEDB database DBD.

SNGL
Indicates that only the selected user partition database is used.

PSELOPT=MULT is the default for the **PSELOPT=** parameter on a XDFLD statement.

REMARKS=
Optional user comments. A 1- to 256-character field.

If your comments contain special characters, such as commas or blank spaces, enclose the full comment string in single quotation marks.

The value specified on the REMARKS keyword cannot contain the following characters:

- Single quotation marks, except when they are used to enclose the full comment string. If a single quotation mark is entered before the end of the full comment string, the remainder of the comment string is truncated. The following examples show correct and incorrect usages of single quotation marks on the REMARKS keyword:

  CORRECT
  ```
  REMARKS='These remarks apply to the XYZ application'
  ```

  INCORRECT
  ```
  REMARKS='These remarks apply to the 'XYZ' application'
  ```
- Double quotation marks.
- Less than (<) symbols.
- Greater than (>) symbols.
- Ampersands (&).

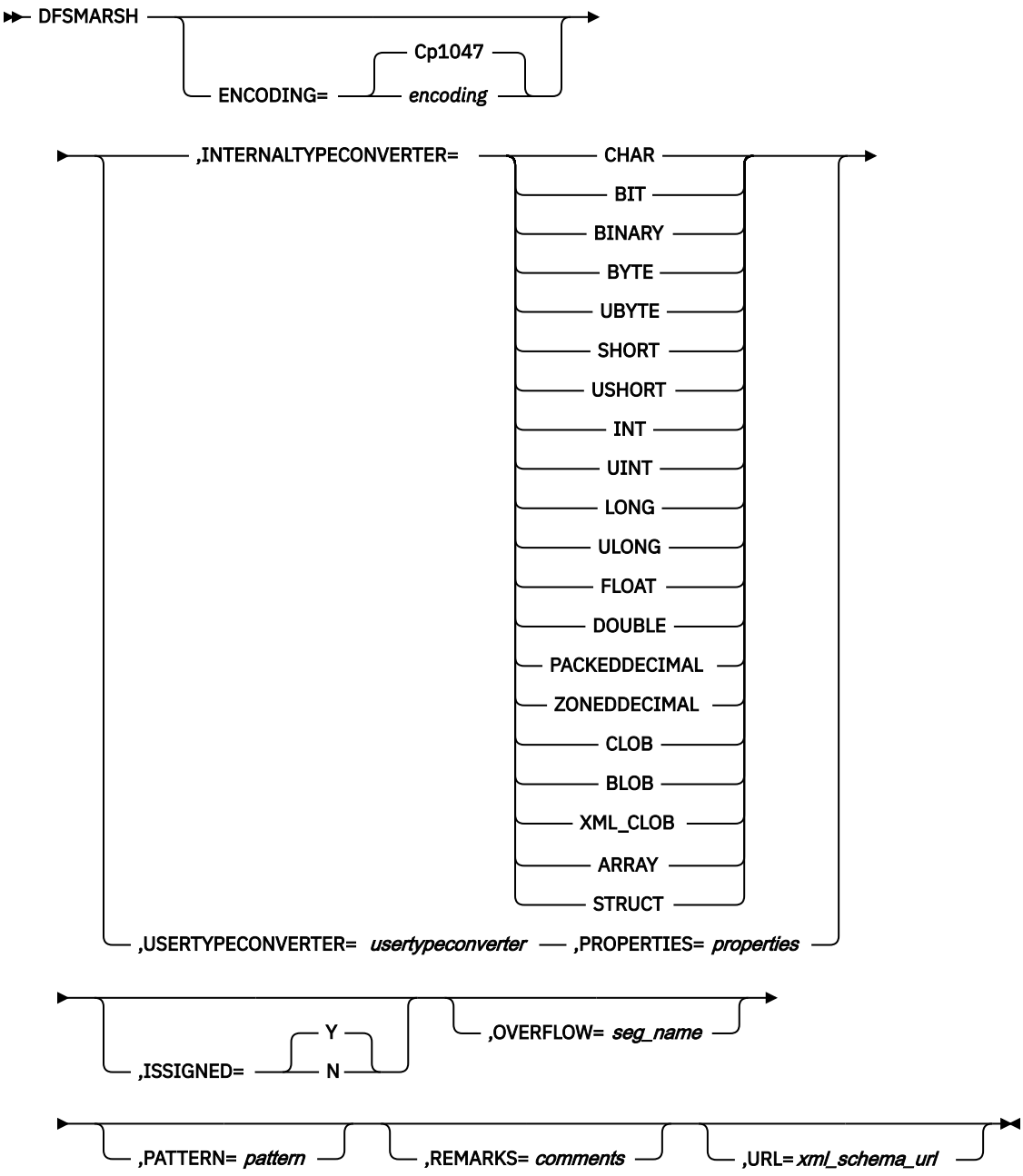
DFSMARSH statements

The DFSMARSH statement defines the marshal attributes for a field.

In the input to the DBD Generation utility, the DFSMARSH statement must immediately follow the FIELD statement to which it applies.

DFSMARSH statement syntax diagram for all database types

The format of the DFSMARSH statement for all database types is shown in the following syntax diagram.



DFSMARSH statement parameter description

ENCODING=

An optional 1- to 25-character field that specifies the encoding of the character data in the field. ENCODING is valid on the DFSMARSH statement only when INTERNALTYPECONVERTER=CHAR.

The value specified on the ENCODING keyword cannot contain the following characters:

- Single and double quotation marks
- Blanks
- Less than (<) and greater than (>) symbols
- Ampersands (&)

If the ENCODING parameter is not specified on the DFSMARSH statement, the default value is determined by the value specified on the ENCODING parameter of either the SEGM statement or, if

ENCODING is not specified on the SEGM statement, the DBD statement. If the ENCODING parameter is not specified on either the SEGM or DBD statement, the default value for the ENCODING parameter is Cp1047, which specifies EBCDIC encoding.

INTERNALTYPECONVERTER=

Specifies the internal conversion routine that IMS uses to convert the IMS data into the data types expected by the application program.

You are required to specify either INTERNALTYPECONVERTER or USERTYPECONVERTER, but not both. INTERNALTYPECONVERTER or USERTYPECONVERTER are mutually exclusive.

When you code the INTERNALTYPECONVERTER parameter on the DFSMARSH statement, you must explicitly code a value. You cannot leave the value blank or specify a null value.

Valid values for the INTERNALTYPECONVERTER parameter are:

ARRAY

BINARY

BIT
If you specify BIT, you must also specify BYTES=1 on the corresponding FIELD statement.

BLOB

BYTE
If you specify BYTE, you must also specify BYTES=1 on the corresponding FIELD statement.

UBYTE
If you specify UBYTE, you must also specify BYTES=1 and either DATATYPE=BYTE or DATATYPE=UBYTE on the corresponding FIELD statement.

CHAR

CLOB

DOUBLE
If you specify DOUBLE, you must also specify BYTES=8 on the corresponding FIELD statement.

FLOAT
If you specify FLOAT, you must also specify BYTES=4 on the corresponding FIELD statement.

INT
If you specify INT, you must also specify BYTES=4 on the corresponding FIELD statement.

UINT
If you specify UBYTE, you must also specify BYTES=4 and either DATATYPE=INT or DATATYPE=UINT on the corresponding FIELD statement.

LONG
If you specify LONG, you must also specify BYTES=8 on the corresponding FIELD statement.

ULONG
If you specify ULONG, you must also specify BYTES=8 and either DATATYPE=LONG or DATATYPE=ULONG on the corresponding FIELD statement.

PACKEDDECIMAL

SHORT
If you specify SHORT, you must also specify BYTES=2 on the corresponding FIELD statement.

USHORT
If you specify USHORT, you must also specify BYTES=2 and either DATATYPE=SHORT or DATATYPE=USHORT on the corresponding FIELD statement.

STRUCT

XML_CLOB

ZONEDDECIMAL

When you specify INTERNALTYPECONVERTER, you must also specify the DATATYPE parameter on the FIELD statement to which this DFSMARSH statement applies.

The value specified on the INTERNALTYPECONVERTER parameter must be consistent with the value specified on the DATATYPE parameter. In most cases, you must specify the same value on INTERNALTYPECONVERTER that you specify on the DATATYPE parameter. The following table shows the valid exceptions to this rule.

Table 9. Additional valid values based on DATATYPE values

DATATYPE value	Valid INTERNALTYPECONVERTER values
BINARY	BINARY, BLOB, CLOB, AND XML_CLOB
BYTE	BYTE, UBYTE
DATE	LONG, CHAR
DECIMAL(*pp*,*ss*)	PACKEDDECIMAL, ZONEDDECIMAL, BINARY
INT	INT, UINT
LONG	LONG, ULONG
SHORT	SHORT, USHORT
TIME	LONG, CHAR
TIMESTAMP	LONG, CHAR
XML	XML_CLOB

If INTERNALTYPECONVERTER=LONG is specified when either DATE, TIME, or TIMESTAMP is specified on the DATATYPE parameter, the value is stored on DASD as the number of milliseconds since January 1, 1970.

If the DFSMARSH statement is not coded, IMS internally sets the value of the INTERNALTYPECONVERTER parameter to a default value determined by the value of the DATATYPE parameter on the field statement.

In most cases, the default value of INTERNALTYPECONVERTER is the same as the value of the DATATYPE parameter. The following table shows the exceptions to this rule.

Table 10. Default values of INTERNALTYPECONVERTER when DFSMARSH statement not specified

DATATYPE value	Default INTERNALTYPECONVERTER value
DATE	LONG
DECIMAL(*pp*,*ss*)	Signed PACKEDDECIMAL
TIME	LONG
TIMESTAMP	LONG
XML	XML_CLOB

ISSIGNED=

Valid only for DATATYPE=DECIMAL.

Valid values are Y or N. The default is Y.

OVERFLOW=

A 1- to 8-character name of a dependent segment that can be used to store any portion of an XML document that does not fit into the field that is defined to hold the XML document.

The parent of the dependent segment is the segment that contains the XML data field. The name of the parent segment must be specified on the PARENT parameter of the SEGM statement that defines the dependent segment.

The OVERFLOW parameter only applies to fields that specify DATATYPE=XML for XML_CLOB data.

PATTERN=

An optional 1- to 50-character field that specifies the pattern to use for the date, time, and timestamp Java datatypes.

The PATTERN parameter applies only when DATE, TIME, or TIMESTAMP is specified on the DATATYPE keyword in the FIELD statement and CHAR is specified on the INTERNALTYPECONVERTER keyword in the DFSMARSH statement. PATTERN is invalid for other datatypes.

Patterns are case sensitive and must be enclosed in single quotation marks.

Except for single quotation marks that are used as delimiters for the keyword value, the value specified on the PATTERN keyword cannot contain the following characters:

- Single and double quotation marks
- Less than (<) and greater than (>) symbols
- Ampersands (&)

The patterns you can specify are defined by the Java class java.text.SimpleDateFormat. The DBD Generation utility does not check that the value entered on PATTERN conforms to the patterns defined by Java.

For example, if you enter the Java format `yyyy.MM.dd`, the resulting time format is "2013.01.01".

PUREDBCS=Y|N

Specifies the use of the character field as the double byte character set.

Valid only for DATATYPE=CHAR. Valid values are Y or N. The default is Y.

PROPERTIES=

Specifies properties for a user type converter specified on the USERTYPECONVERTER parameter. These properties are passed to the user type converter.

Valid only when USERTYPECONVERTER is specified.

The names and properties specified on the PROPERTIES keyword are case sensitive.

The following characters are not supported by the PROPERTIES keyword:

- Single and double quotation marks
- Blanks
- Less than (<) and greater than (>) symbols
- Ampersands (&)

The maximum length for a property name is 128 characters. The maximum length for a property value is also 128 characters.

The format is:

```
PROPERTIES=(property1_name=property1_value,property2_name=property2_value)
```

For example,

```
PROPERTIES=(DOG=BUTCH,CAT=LUCY)
```

REMARKS=

Optional user comments. A 1- to 256-character field.

If your comments contain special characters, such as commas or blank spaces, enclose the full comment string in single quotation marks.

The value specified on the REMARKS keyword cannot contain the following characters:

- Single quotation marks, except when they are used to enclose the full comment string. If a single quotation mark is entered before the end of the full comment string, the remainder of the comment

string is truncated. The following examples show correct and incorrect usages of single quotation marks on the REMARKS keyword:

CORRECT

```
REMARKS='These remarks apply to the XYZ application'
```

INCORRECT

```
REMARKS='These remarks apply to the 'XYZ' application'
```

- Double quotation marks.
- Less than (<) symbols.
- Greater than (>) symbols.
- Ampersands (&).

URL=

An optional 1- to 256-character field for the URL that references the XML schema that describes this field. For example, URL=MySchema.xsd.

The value specified on the URL keyword cannot contain the following characters:

- Single and double quotation marks
- Blanks
- Less than (<) and greater than (>) symbols
- Ampersands (&)

The URL parameter applies only to fields that specify DATATYPE=XML for XML_CLOB data.

USERTYPECONVERTER=

A 1- to 256-character, fully-qualified Java class name of the user provided Java class to be used for type conversion.

The value specified on the USERTYPECONVERTER keyword cannot contain the following characters:

- Single and double quotation marks
- Blanks
- Less than (<) and greater than (>) symbols
- Ampersands (&)

Mutually exclusive with INTERNALTYPECONVERTER.

For example,

```
USERTYPECONVERTER=class://com.ibm.ims.dli.types.PackedDateConverter
```

Examples of the DFSMARSH statement

The following series of examples show some possible uses of the DFSMARSH statement for various DATATYPE and type converter specifications.

DATATYPE=DATE:

```
FIELD    EXTERNALNAME=XDATE,
         BYTES=8,
         START=84,
         DATATYPE=DATE
DFSMARSH ENCODING=Cp1047,
         INTERNALTYPECONVERTER=CHAR,
         PATTERN='MMddyyyy'
```

DATATYPE=TIME:

```
FIELD    EXTERNALNAME=XTIME,
         BYTES=6,
         START=92,
         DATATYPE=TIME
```

```
DFSMARSH ENCODING=Cp1047,
         INTERNALTYPECONVERTER=CHAR,
         PATTERN='HHmmss'
```

DATATYPE=TIMESTAMP:

```
FIELD    EXTERNALNAME=XTIMESTAMP,
         BYTES=16,
         START=84,
         DATATYPE=TIMESTAMP
DFSMARSH ENCODING=Cp1047,
         INTERNALTYPECONVERTER=CHAR,
         PATTERN='MMddyyyyHHmmssff'
```

DATATYPE=ZONEDDECIMAL:

```
FIELD NAME=ORDPRICE,
         BYTES=10,
         START=21,
         DATATYPE=DECIMAL(10,2)
DFSMARSH INTERNALTYPECONVERTER=ZONEDDECIMAL,
         ISSIGNED=Y
```

DATATYPE=PACKEDDECIMAL:

```
FIELD    EXTERNALNAME=XPACKEDDEC1,
         BYTES=4,
         START=60,
         DATATYPE=DECIMAL(7,2)
DFSMARSH INTERNALTYPECONVERTER=PACKEDDECIMAL,
         ISSIGNED=Y
```

USERTYPECONVERTER=:

```
FIELD    EXTERNALNAME=PACKEDDATEFIELD,
         BYTES=5,
         START=40,
         DATATYPE=DATE
DFSMARSH USERTYPECONVERTER=class://com.ibm.ims.dli.types.PackedDateConverter,
         PROPERTIES=(pattern=MMddyyyy,isSigned=false)
```

Related tasks

Specifying data types for application programs (Database Administration)

Related reference

"FIELD statements" on page 101

The FIELD statement defines a field within a segment type. Fields are referred to by PSBs when defining sensitivity to the fields or by an application program in a DL/I call segment search argument.

Related information

com.ibm.ims.dli.types Package Javadoc

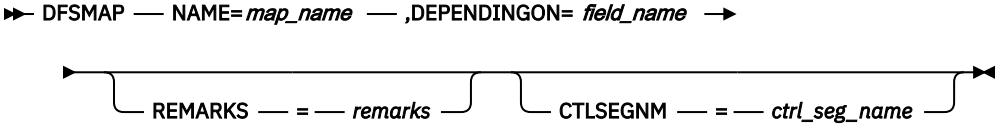
DFSMAP statements

The DFSMAP statement enables the alternate mapping of fields within a segment.

The DFSMAP statement defines a group of one or more map cases and relates the cases to a control field. The control field identifies which map case is used in a given segment instance.

The format of the DFSMAP statement for all database types is shown in the following syntax diagram.

DFSMAP statement syntax diagram for all database types



DFSMAP statement parameter description

CTLSEGNM=
An optional parameter that specifies the NAME of the segment whose key feedback data is used to determine which map case is used for a given segment instance. The segment specified must be in the hierarchical path of the current segment for its value to be stored in the IMS catalog. When the CTLSEGNM parameter is specified, the DEPENDINGON= parameter must specify the name of the control field that is within the key range of the segment that is specified in CTLSEGNM.

The CTLSEGNM= specifies a name as a 1 to 8-character uppercase alphanumeric string. Each character must be in the range of A through Z, or 0 through 9, or be the character $, #, @. The first character cannot be numeric. The same restrictions apply as those for the NAME= parameter of the SEGM statement. The specified segment must be in the hierarchical path of the current segment.

Note:

- CTLSEGNM cannot be specified for GSAM, MSDB, logical and index databases.
- The CTLSEGNM parameter is ignored if the name of the segment specified is blank or if it is the name of the current segment.

DEPENDINGON=
The external name of the control field within this segment that contains the value that determines which map case is used for a given segment instance.

When the CTLSEGNM parameter is also specified, the external name of the control field must be a field within the key range of the segment specified in CTLSEGNM. Otherwise, the external name of the control field must be within this segment. When the CTLSEGNM parameter is also specified, the validation of DEPENDINGON is deferred during DFSMAP processing and is completed after all the SEGM statements are processed. If the control field does not contain a value that corresponds to a CASEID in a DFSCASE statement for this map, this map is not used for this segment instance. If the FIELD statement that defines the control field does not explicitly code the EXTERNALNAME parameter, specify the value of the NAME parameter in the DEPENDINGON field.

NAME=
A required 1- to 128-character alphanumeric field that defines the name of this map. Blanks are not supported.

REMARKS=
Optional user comments. A 1- to 256-character field.

If your comments contain special characters, such as commas or blank spaces, enclose the full comment string in single quotation marks.

The value specified on the REMARKS keyword cannot contain the following characters:

- Single quotation marks, except when they are used to enclose the full comment string. If a single quotation mark is entered before the end of the full comment string, the remainder of the comment string is truncated. The following examples show correct and incorrect usages of single quotation marks on the REMARKS keyword:

  CORRECT
  ```
  REMARKS='These remarks apply to the XYZ application'
  ```

  INCORRECT
  ```
  REMARKS='These remarks apply to the 'XYZ' application'
  ```
- Double quotation marks.
- Less than (<) symbols.
- Greater than (>) symbols.
- Ampersands (&).

Mapping example: DFSMAP and DFSCASE

The following example shows how mapping might be used in the DBD source to define a single segment that is used to store data about three different types of insurance policy: an auto insurance policy, a home insurance policy, and a boat insurance policy. Each policy type requires different fields to hold the information that is unique to that policy type.

In the DBD source, the fields for each policy type are mapped by a different DFSCASE statement. The three map cases in the example are named AUTOMAP, HOMEMAP, and BOATMAP. The fields that make up the map defined by a given DFSCASE statement each specify the name of the DFSCASE statement that they belong to on the CASENAME parameter in their FIELD statement. The DFSCASE statements are grouped by the DFSMAP statement POLICYMAPS in the segment CUSTOMERPOLICY.

The value specified on the CASEID parameter of each map case uniquely identifies the map case and serves as the control field value. When a segment instance is first inserted into the database, the ID of the map case that the segment instance uses is inserted into the control field. In the example, the control field is named POLICYTYPE. At run time, when an application program retrieves the segment from the database, the application program must evaluate the control field value to determine the correct mapping of the fields.

```
  DBD       NAME=POLICYDB,                                          C
               ENCODING=CP1047,                                     C
               ACCESS=(DEDB),                                       C
               RMNAME=(RMOD3),                                      C
               PASSWD=NO
       AREA    DD1=PLCYAR01,                                        C
               DEVICE=3330,                                         C
               SIZE=(2048),                                         C
               UOW=(15,10),                                         C
               ROOT=(10,5),                                         C
               REMARKS='AREA NUMBER 1 FOR POLICYDB DATABASE'
    SEGM       NAME=CUSTOMER,                                       C
               PARENT=0,                                            C
               BYTES=(390,20)
      FIELD    NAME=(CUSTKEY,SEQ,U),                                C
               BYTES=12,                                            C
               START=1,                                             C
               TYPE=C
   SEGM    NAME=POLICY,                                            C
               EXTERNALNAME=CUSTOMERPOLICY,                         C
               ENCODING=CP1047,                                     C
               PARENT=CUSTOMER,                                     C
               BYTES=(900),                                         C
               TYPE=DIR,                                            C
               RULES=(LLL,HERE)
*********************************************************************
*        CONTROL FIELD:
*********************************************************************
      FIELD    EXTERNALNAME=POLICYTYPE,                             C
               BYTES=4,                                             C
               START=1,                                             C
               DATATYPE=CHAR
*********************************************************************
*        DFSMAP STATEMENT:
*********************************************************************
      DFSMAP   NAME=POLICYMAPS,                                     C
               DEPENDINGON=POLICYTYPE
*********************************************************************
*        DFSCASE STATEMENT 1:
*********************************************************************
      DFSCASE  NAME=AUTOMAP,                                        C
               CASEID=AUTO,                                         C
               CASEIDTYPE=C,                                        C
               MAPNAME=POLICYMAPS,                                  C
               REMARKS='DEFINES THE FIELDS OF AN AUTO INSURANCE POLICY'
      FIELD    EXTERNALNAME=AUTOMAKE,                               C
               CASENAME=AUTOMAP,                                    C
               BYTES=15,                                            C
               START=5,                                             C
               DATATYPE=CHAR
      FIELD    EXTERNALNAME=MODEL,                                  C
               CASENAME=AUTOMAP,                                    C
               BYTES=15,                                            C
               START=20,                                            C
               DATATYPE=CHAR
```

```
      FIELD    EXTERNALNAME=YEAR,                                      C
               CASENAME=AUTOMAP,                                       C
               BYTES=4,                                                C
               START=35,                                               C
               DATATYPE=CHAR
*********************************************************************
*        DFSCASE STATEMENT 2:
*********************************************************************
      DFSCASE  NAME=HOMEMAP,                                           C
               CASEID=HOME,                                            C
               CASEIDTYPE=C,                                           C
               MAPNAME=POLICYMAPS,                                     C
               REMARKS='DEFINES THE FIELDS OF A HOME INSURANCE POLICY'
      FIELD    EXTERNALNAME=DWELLING_TYPE,                             C
               CASENAME=HOMEMAP,                                       C
               BYTES=20,                                               C
               START=5,                                                C
               DATATYPE=CHAR
      FIELD    EXTERNALNAME=ROOMS,                                     C
               CASENAME=HOMEMAP,                                       C
               BYTES=5,                                                C
               START=25,                                               C
               DATATYPE=CHAR
      FIELD    EXTERNALNAME=SQ_FOOT,                                   C
               CASENAME=HOMEMAP,                                       C
               BYTES=6,                                                C
               START=30,                                               C
               DATATYPE=CHAR
*********************************************************************
*        DFSCASE STATEMENT 3:
*********************************************************************
      DFSCASE  NAME=BOATMAP,                                           C
               CASEID=BOAT,                                            C
               CASEIDTYPE=C,                                           C
               MAPNAME=POLICYMAPS,                                     C
               REMARKS='DEFINES THE FIELDS OF A BOAT INSURANCE POLICY'
      FIELD    EXTERNALNAME=CLASS,                                     C
               CASENAME=BOATMAP,                                       C
               BYTES=10,                                               C
               START=5,                                                C
               DATATYPE=CHAR
      FIELD    EXTERNALNAME=LENGTH,                                    C
               CASENAME=BOATMAP,                                       C
               BYTES=6,                                                C
               START=15,                                               C
               DATATYPE=CHAR
      FIELD    EXTERNALNAME=BOATMAKE,                                  C
               CASENAME=BOATMAP,                                       C
               BYTES=10,                                               C
               START=21,                                               C
               DATATYPE=CHAR
   DBDGEN
   FINISH
   END
```

Related tasks

Defining alternative field maps for a segment (Database Administration)

Related reference

“FIELD statements” on page 101

The FIELD statement defines a field within a segment type. Fields are referred to by PSBs when defining sensitivity to the fields or by an application program in a DL/I call segment search argument.

“DFSCASE statements” on page 136

The DFSCASE statement defines a *map case*, which is a set of FIELD statements that together define an optional, alternative field layout for a given byte range within a segment definition.

DFSCASE statements

The DFSCASE statement defines a *map case*, which is a set of FIELD statements that together define an optional, alternative field layout for a given byte range within a segment definition.

Map cases that map the same byte range in a segment are grouped by a DFSMAP statement. The DFSMAP statement also links the map cases to a separately defined control field in the segment definition.

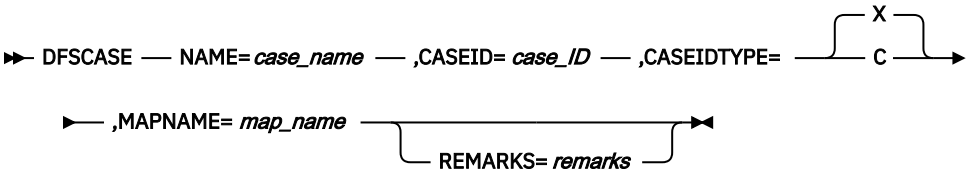
Each map case has a unique ID. In an instance of the segment, the ID of the map case that is in effect is stored in the control field when the segment is created.

Unless the IMS universal drivers are used, the field layouts that are defined by the map cases must be defined to the application programs that access this byte range by a COBOL copybook or other programming artifact. At the time a segment instance is accessed, the application programs determine which copybook to use by checking the value of the control field.

When application programs access IMS through the IMS Universal drivers, no additional programming artifacts are needed to define the field layouts to the application programs.

The format of the DFSCASE statement for all database types is shown in the following syntax diagram.

DFSCASE statement syntax diagram for all database types



DFSCASE statement parameter description

CASEID
A 1- to 128-byte field that defines a unique identifier for the case.

A segment instance specifies the CASEID value in a user-defined control field when part or all of the field structure of the segment is mapped by this case.

When CASEIDTYPE=C, the CASEID field can contain alphanumeric characters, _, @, $, and #. Single quotation marks are supported, but not required. Blanks are not supported.

When CASEIDTYPE=X, the only valid characters in the CASEID parameter are 0-9 and A-F.

The length of the CASEID value must be supported by the length of the user-defined control field. If CASEIDTYPE=C, the length of the CASEID value must be less than or equal to the value specified on the BYTES parameter of the control field. If CASEIDTYPE=X, the length of the CASEID value must be exactly equal to twice the value specified on the BYTES parameter of the control field.

A case ID must be unique within the map that the case belongs to.

CASEIDTYPE
Defines the data type of the value specified in the CASEID parameter. Valid values are C, which specifies Cp1047 (EBCDIC character encoding) and X, which specifies hexadecimal.

Depending on whether C or X is specified on CASEIDTYPE, the valid length of the CASEID value is calculated differently. The length is valid when it is consistent with the length specified on the BYTES parameter of the field referenced by the DEPENDINGON parameter in the DFSMAP statement. For CASEIDTYPE=C, the length of the CASEID value must be less than or equal to the value specified on the BYTES parameter. For CASEIDTYPE=X, the length of the CASEID value must be exactly equal to twice the value specified on the BYTES parameter.

MAPNAME
The name of the map that this case belongs to, as specified on the NAME parameter in the DFSMAP statement. This field is required.

NAME
A required 1- to 128-character, alphanumeric field that defines the name of this case. Blanks are not supported.

A case name must be unique within a segment.

REMARKS=

Optional user comments. A 1- to 256-character field.

If your comments contain special characters, such as commas or blank spaces, enclose the full comment string in single quotation marks.

The value specified on the REMARKS keyword cannot contain the following characters:

- Single quotation marks, except when they are used to enclose the full comment string. If a single quotation mark is entered before the end of the full comment string, the remainder of the comment string is truncated. The following examples show correct and incorrect usages of single quotation marks on the REMARKS keyword:

  CORRECT

  ```
  REMARKS='These remarks apply to the XYZ application'
  ```

  INCORRECT

  ```
  REMARKS='These remarks apply to the 'XYZ' application'
  ```

- Double quotation marks.
- Less than (<) symbols.
- Greater than (>) symbols.
- Ampersands (&).

Mapping example: DFSMAP and DFSCASE

The following example shows how mapping might be used in the DBD source to define a single segment that is used to store data about three different types of insurance policy: an auto insurance policy, a home insurance policy, and a boat insurance policy. Each policy type requires different fields to hold the information that is unique to that policy type.

In the DBD source, the fields for each policy type are mapped by a different DFSCASE statement. The three map cases in the example are named AUTOMAP, HOMEMAP, and BOATMAP. The fields that make up the map defined by a given DFSCASE statement each specify the name of the DFSCASE statement that they belong to on the CASENAME parameter in their FIELD statement. The DFSCASE statements are grouped by the DFSMAP statement POLICYMAPS in the segment CUSTOMERPOLICY.

The value specified on the CASEID parameter of each map case uniquely identifies the map case and serves as the control field value. When a segment instance is first inserted into the database, the ID of the map case that the segment instance uses is inserted into the control field. In the example, the control field is named POLICYTYPE. At run time, when an application program retrieves the segment from the database, the application program must evaluate the control field value to determine the correct mapping of the fields.

```
DBD       NAME=POLICYDB,                                          C
             ENCODING=CP1047,                                     C
             ACCESS=(DEDB),                                       C
             RMNAME=(RMOD3),                                      C
             PASSWD=NO
     AREA    DD1=PLCYAR01,                                        C
             DEVICE=3330,                                         C
             SIZE=(2048),                                         C
             UOW=(15,10),                                         C
             ROOT=(10,5),                                         C
             REMARKS='AREA NUMBER 1 FOR POLICYDB DATABASE'
  SEGM       NAME=CUSTOMER,                                       C
             PARENT=0,                                            C
             BYTES=(390,20)
    FIELD    NAME=(CUSTKEY,SEQ,U),                                C
             BYTES=12,                                            C
             START=1,                                             C
             TYPE=C
 SEGM    NAME=POLICY,                                            C
             EXTERNALNAME=CUSTOMERPOLICY,                         C
             ENCODING=CP1047,                                     C
             PARENT=CUSTOMER,                                     C
             BYTES=(900),                                         C
             TYPE=DIR,                                            C
             RULES=(LLL,HERE)
```

```
*********************************************************************
*        CONTROL FIELD:
*********************************************************************
      FIELD    EXTERNALNAME=POLICYTYPE,                                C
               BYTES=4,                                                C
               START=1,                                                C
               DATATYPE=CHAR
*********************************************************************
*        DFSMAP STATEMENT:
*********************************************************************
      DFSMAP   NAME=POLICYMAPS,                                        C
               DEPENDINGON=POLICYTYPE
*********************************************************************
*        DFSCASE STATEMENT 1:
*********************************************************************
      DFSCASE  NAME=AUTOMAP,                                           C
               CASEID=AUTO,                                            C
               CASEIDTYPE=C,                                           C
               MAPNAME=POLICYMAPS,                                     C
               REMARKS='DEFINES THE FIELDS OF AN AUTO INSURANCE POLICY'
      FIELD    EXTERNALNAME=AUTOMAKE,                                  C
               CASENAME=AUTOMAP,                                       C
               BYTES=15,                                               C
               START=5,                                                C
               DATATYPE=CHAR
      FIELD    EXTERNALNAME=MODEL,                                     C
               CASENAME=AUTOMAP,                                       C
               BYTES=15,                                               C
               START=20,                                               C
               DATATYPE=CHAR
      FIELD    EXTERNALNAME=YEAR,                                      C
               CASENAME=AUTOMAP,                                       C
               BYTES=4,                                                C
               START=35,                                               C
               DATATYPE=CHAR
*********************************************************************
*        DFSCASE STATEMENT 2:
*********************************************************************
      DFSCASE  NAME=HOMEMAP,                                           C
               CASEID=HOME,                                            C
               CASEIDTYPE=C,                                           C
               MAPNAME=POLICYMAPS,                                     C
               REMARKS='DEFINES THE FIELDS OF A HOME INSURANCE POLICY'
      FIELD    EXTERNALNAME=DWELLING_TYPE,                             C
               CASENAME=HOMEMAP,                                       C
               BYTES=20,                                               C
               START=5,                                                C
               DATATYPE=CHAR
      FIELD    EXTERNALNAME=ROOMS,                                     C
               CASENAME=HOMEMAP,                                       C
               BYTES=5,                                                C
               START=25,                                               C
               DATATYPE=CHAR
      FIELD    EXTERNALNAME=SQ_FOOT,                                   C
               CASENAME=HOMEMAP,                                       C
               BYTES=6,                                                C
               START=30,                                               C
               DATATYPE=CHAR
*********************************************************************
*        DFSCASE STATEMENT 3:
*********************************************************************
      DFSCASE  NAME=BOATMAP,                                           C
               CASEID=BOAT,                                            C
               CASEIDTYPE=C,                                           C
               MAPNAME=POLICYMAPS,                                     C
               REMARKS='DEFINES THE FIELDS OF A BOAT INSURANCE POLICY'
      FIELD    EXTERNALNAME=CLASS,                                     C
               CASENAME=BOATMAP,                                       C
               BYTES=10,                                               C
               START=5,                                                C
               DATATYPE=CHAR
      FIELD    EXTERNALNAME=LENGTH,                                    C
               CASENAME=BOATMAP,                                       C
               BYTES=6,                                                C
               START=15,                                               C
               DATATYPE=CHAR
      FIELD    EXTERNALNAME=BOATMAKE,                                  C
               CASENAME=BOATMAP,                                       C
               BYTES=10,                                               C
               START=21,                                               C
               DATATYPE=CHAR
   DBDGEN
```

```
    FINISH
    END
```

Related tasks

Defining alternative field maps for a segment (Database Administration)

Related reference

"FIELD statements" on page 101
The FIELD statement defines a field within a segment type. Fields are referred to by PSBs when defining sensitivity to the fields or by an application program in a DL/I call segment search argument.

"DFSMAP statements" on page 133
The DFSMAP statement enables the alternate mapping of fields within a segment.

DBDGEN, FINISH, and END statements

All DBD generation utility control statements must be followed by an END statement.

There are three additional utility statements. Two are required (DBDGEN and END) and one is optional (FINISH).

The DBDGEN statement indicates the end of DBD generation statements used to define the DBD. This statement is required. The following example shows the format of the DBDGEN statement for all database types.

The FINISH statement is optional and is retained for compatibility. The following example shows the format of the FINISH statement for all database types.

The END statement indicates the end of input statements to the assembler. This statement is required. The following example shows the format of the END statement for all database types.

Examples of the DBDGEN utility

These examples show how to use the DBDGEN utility for DBD generation for different database types.

An application program through a database PCB can operate on any of the databases previously described. The value of the DBDNAME= parameter on the database statement should equal the value of the NAME= parameter on a DBD statement of DBD generation. The SENSEG statements following the database statements in PSB generation should reference segments defined by SEGM statements in the named DBD generation.

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

Examples without secondary indexes or logical relationships

The DBD generation examples show the statements that are required to define HSAM, HISAM, HDAM, HIDAM, primary HIDAM index, GSAM, MSDB, and DEDB databases without secondary indexes or logical relationships.

Two data structures are shown in the following figure. One represents the hierarchic order of data used in a payroll inventory data structure, which includes NAME, ADDRESS, and PAYROLL. The other structure

represents the hierarchic order of data used in a skills inventory data structure, which includes SKILL, NAME, EXPERIENCE, and EDUCATION.

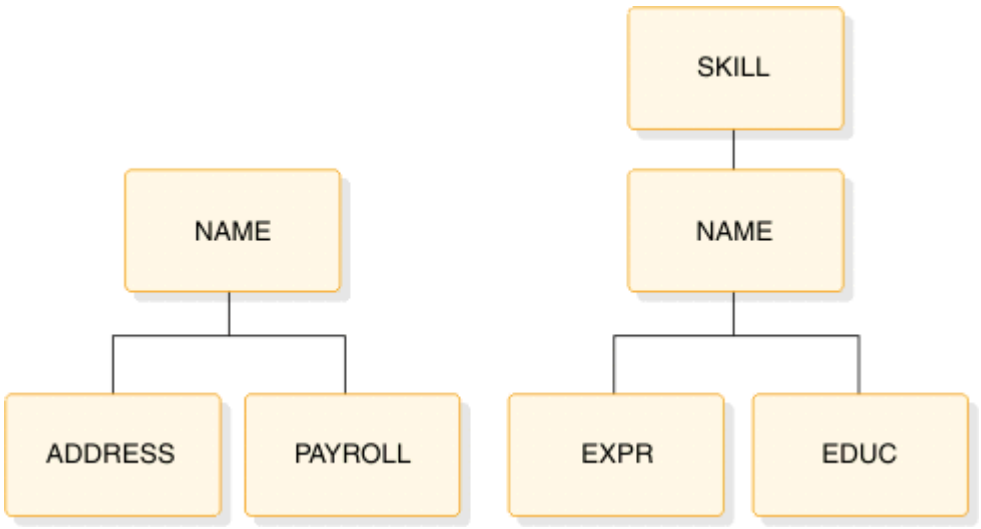


Figure 4. Payroll and skills inventory data structures

HSAM DBD generation example

The following examples show the DBD generation statements that define the skills inventory and payroll data structures as HSAM databases.

HSAM DBD generation of skills inventory database

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

SEGM  NAME=SKILL,BYTES=31,FREQ=100, PARENT=0
FIELD NAME=TYPE,BYTES=21,START=1,TYPE=C
FIELD NAME=STDCODE,BYTES=10,START=22,TYPE=C

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

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

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

DBDGEN
FINISH
END
```

HSAM DBD generation of payroll database

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

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

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

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

```
DBDGEN
FINISH
END
```

HISAM DBD generation example

The following examples show the DBD generation statements that define the skills inventory and payroll data structures as HISAM databases.

HISAM DBD generation of skills inventory SKILLINV database

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

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

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

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

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

DBDGEN
FINISH
END
```

HISAM DBD generation of payroll database

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

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

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

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

DBDGEN
FINISH
END
```

HDAM DBD generation example

The following examples show the statements required to define the skills inventory data structure as HDAM databases. The first example defines a database that uses hierarchic pointers, and the second example defines a database that uses physical child and physical twin pointers. The third example defines a database that uses the VERSION= and EXIT= parameters.

HDAM DBD generation of skills inventory SKILLINV database with hierarchic pointers

```
DBD    NAME=SKILLINV,ACCESS=HDAM,RMNAME=(DFSHDC40,20,500,824)
DATASET    DD1=SKILHDAM,BLOCK=4096,SCAN=0

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

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

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

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

DBDGEN
FINISH
END
```

HDAM DBD generation of skills inventory database with physical child and physical twin pointers

```
DBD    NAME=SKILLINV,ACCESS=HDAM,RMNAME=(DFSHDC40,20,500,824)
DATASET    DD1=SKILHDAM,BLOCK=4096,SCAN=0

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

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

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

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

DBDGEN
FINISH
END
```

HDAM DBD generation of skills inventory SKILLINV database with EXIT= and VERSION= parameters

```
DBD    NAME=SKILLINV,ACCESS=HDAM,RMNAME=(DFSHDC40,20,500,824),VERSION=CCCCCC
DATASET    DD1=SKILHDAM,BLOCK=4096,SCAN=0

SEGM   NAME=A,BYTES=8,PTR=H,PARENT=0,EXIT=(EXITA)
FIELD  NAME=(TYPE,SEQ,U),BYTES=21,START=1,TYPE=C
FIELD  NAME=STDCODE,BYTES=10,START=22,TYPE=C

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

SEGM   NAME=C,BYTES=8,PTR=H,PARENT=A,EXIT=((EXITA,PATH),(EXITC))
FIELD  NAME=PREVJOB,BYTES=10,START=1,TYPE=C
FIELD  NAME=CLASSIF,BYTES=10,START=11,TYPE=C

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

DBDGEN
FINISH
END
```

HIDAM DBD generation example

A HIDAM database is indexed through the sequence field of its root segment type. In defining the HIDAM and primary HIDAM index databases, an index relationship is established between the HIDAM root segment type and the segment type defined in the primary HIDAM index database. The following examples summarize the statements required to establish the index relationship between the HIDAM root segment type and the index segment type in the primary HIDAM index database. Only those operands pertinent to the index relationship are shown.

Primary HIDAM index relationship

```
HIDAM:                          INDEX:

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

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

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

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

The following examples show the statements that define the skills inventory data structure as two HIDAM databases. The first is defined with hierarchic pointers, and the second is defined with physical child and physical twin pointers. Since a HIDAM database is indexed on the sequence field of its root segment type, an INDEX DBD generation is required. The following example shows the statements for the two HIDAM DBD generations and the index DBD generation.

INDEX DBD generation for HIDAM database SKILLINV

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

HIDAM DBD generation of skills inventory database with hierarchic pointers

```
DBD    NAME=SKILLINV,ACCESS=HIDAM
DATASET   DD1=SKLHIDAM,BLOCK=4096,SCAN=0

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

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

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

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

DBDGEN
FINISH
END
```

HIDAM DBD generation of skills inventory SKILLINV database with physical child and physical twin pointers

```
DBD    NAME=SKILLINV,ACCESS=HIDAM
DATASET   DD1=SKLHIDAM,BLOCK=4096,SCAN=0

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

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

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

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

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

DBDGEN
```

```
FINISH
END
```

PHDAM DBD generation example

The following example shows the DBD generation of skills inventory database with physical child and physical twin pointers for a PHDAM database.

```
 DBD    NAME=SKILLINV,ACCESS=(PHDAM,OSAM),RMNAME=(DFSHDC40,20,500,824)
 SEGM   NAME=SKILL,BYTES=31,PTR=T,PARENT=0
 FIELD  NAME=(TYPE,SEQ,U),BYTES=21,START=1,TYPE=C
 FIELD  NAME=STDCODE,BYTES=10,START=22,TYPE=C
 SEGM   NAME=NAME,BYTES=20,PTR=T,PARENT=((SKILL,SNGL))

 FIELD  NAME=(STDCLEVL,SEQ,U),BYTES=20,START=1,TYPE=C
 SEGM   NAME=EXPR,BYTES=20,PTR=T,PARENT=((NAME,SNGL))
 FIELD  NAME=PREVJOB,BYTES=10,START=1,TYPE=C
 FIELD  NAME=CLASSIF,BYTES=10,START=11,TYPE=C
 SEGM   NAME=EDUC,BYTES=75,PTR=T,PARENT=((NAME,SNGL))
 FIELD  NAME=GRADLEVL,BYTES=10,START=1,TYPE=C
 FIELD  NAME=SCHOOL,BYTES=65,START=11,TYPE=C
 DBDGEN
 END
```

PHIDAM DBD generation example

The following example shows the DBD generation of skills inventory database with physical child and physical twin pointers for a PHIDAM database. No index base definitions are required.

```
  DBD    NAME=SKILLINV,ACCESS=PHIDAM
  SEGM  NAME=SKILL,BYTES=31,PTR=T,PARENT=0
  FIELD NAME=(TYPE,SEQ,U),BYTES=21,START=1,TYPE=C
  FIELD NAME=STDCODE,BYTES=10,START=22,TYPE=C
  SEGM   NAME=NAME,BYTES=20,PTR=T,PARENT=((SKILL,SNGL))
  FIELD NAME=(STDCLEVL,SEQ,U),BYTES=20,START=1,TYPE=C
  SEGM  NAME=EXPR,BYTES=20,PTR=T,PARENT=((NAME,SNGL))
  FIELD NAME=PREVJOB,BYTES=10,START=1,TYPE=C
  FIELD NAME=CLASSIF,BYTES=10,START=11,TYPE=C
  SEGM  NAME=EDUC,BYTES=75,PTR=T,PARENT=((NAME,SNGL))
  FIELD NAME=GRADLEVL,BYTES=10,START=1,TYPE=C
  FIELD NAME=SCHOOL,BYTES=65,START=11,TYPE=C
  DBDGEN
  FINISH
  END
```

GSAM DBD generation example

The following example shows the DBD generation statements that define input and output data sets for a GSAM database.

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

MSDB DBD generation examples

The following examples show the DBD generation statements necessary to define the three types of main storage database DBDs.

DBD generation for a nonterminal-related MSDB without LTERM keys

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

DBD generation for a nonterminal-related MSDB with LTERM keys

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

DBD generation for a fixed terminal-related MSDB

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

DBD generation for a dynamic terminal-related MSDB

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

Data entry database DBD generations example

The following example shows the DBD generation statements necessary to define a data entry database DBD.

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

DBD generation of DEDB subset pointers example

The following example shows the DBD generation statements necessary to define a DEDB with subset pointers.

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

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

Examples with logical relationships

You can define three types of logical relationships: unidirectional, bidirectional physically paired, and bidirectional virtually paired.

The following figure shows the three types of logical relationships that can be defined in IMS databases. The tables that follow the figure define the statements that are required to define each type of relationship. Only the operands pertinent to the relationship are shown, and it is assumed that each type of relationship is defined between segments in two databases named DBD1 and DBD2.

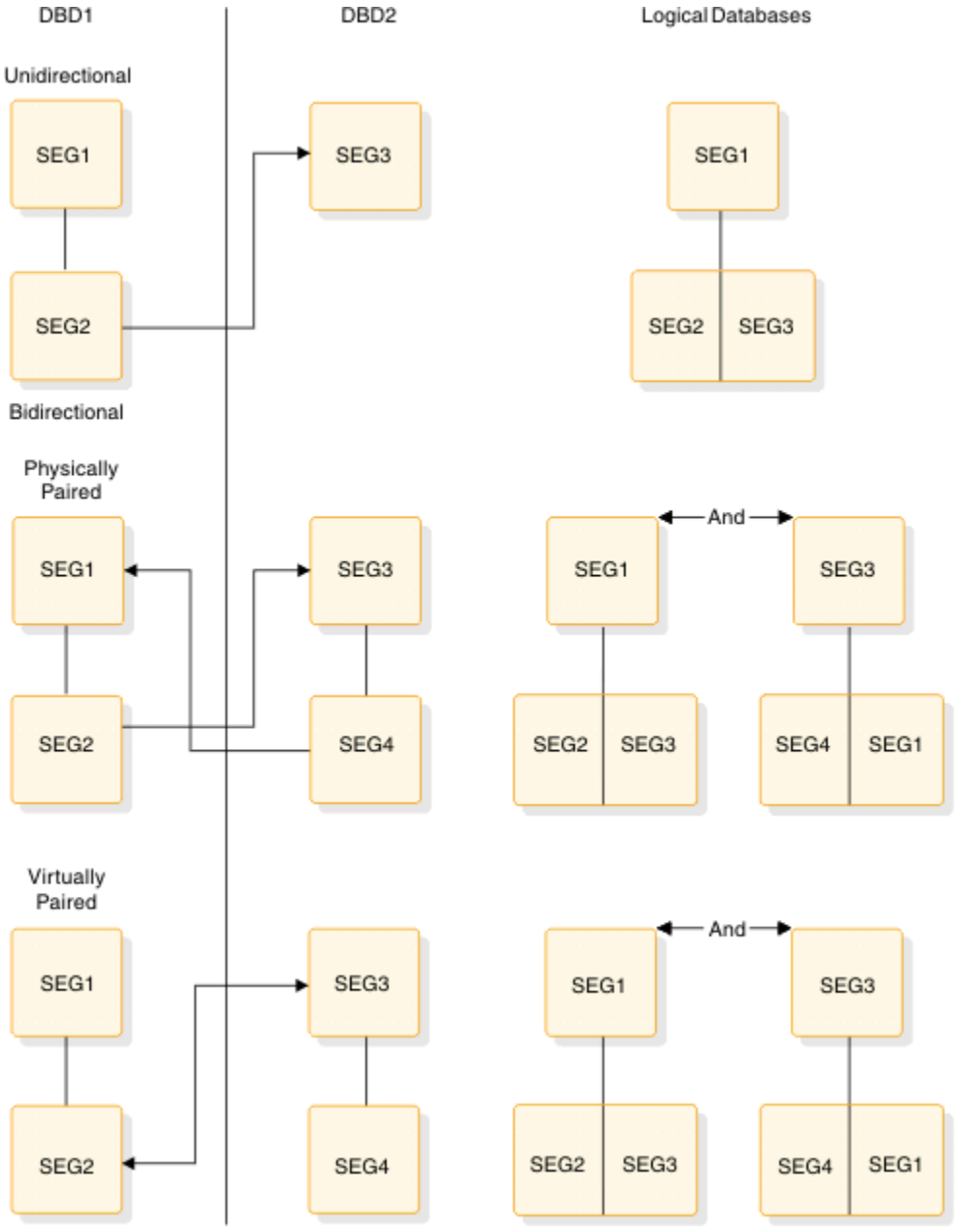


Figure 5. Comparison of unidirectional, physically paired bidirectional, and virtually paired bidirectional logical relationships

The following tables show statements that are required to define each type of relationship. Only the operands pertinent to the relationship are shown, and it is assumed that each type of relationship is defined between segments in two databases named DBD1 and DBD2.

Table 11. Statements that are required to define unidirectional logical relationships

Statements for DBD1	Statements for DBD2
SEGM NAME=SEG1,PARENT= ,BYTES=,FREQ= ,POINTER=,RULES=	SEGM NAME=SEG3,PARENT= ,BYTES=,FREQ=,POINTER= ,RULES=
SEGM NAME=SEG2 ,PARENT=((SEG1,) ,SEG3,PHYSICAL,DBD2))[1] ,BYTES=,FREQ= ,POINTER=(LPARNT)[1] ,RULES=	LCHILD NAME=(SEG2,DBD1)

Note:

1. Specify symbolic or direct logical parent pointer. The direct-access pointer can be specified only when the logical parent is in an HDAM, HIDAM, PHDAM or HIDAM database.

Table 12. Statements that are required to define physically paired bidirectional logical relationships

Statements for DBD1	Statements for DBD2
SEGM NAME=SEG1,PARENT= ,BYTES=,FREQ,= ,POINTER=,RULES=	SEGM NAME=SEG3,PARENT= ,BYTES=,FREQ= ,POINTER=,RULES=
LCHILD NAME=(SEG4,DBD2) ,PAIR=SEG2	LCHILD NAME=(SEG2,DBD1) ,PAIR=SEG4
SEGM NAME=SEG2 ,PARENT=((SEG1,) ,(SEG3,PHYSICAL,DBD2))[1] ,BYTES=,FREQ= ,POINTER=(LPARNT,PAIRED)[1] ,RULES=	SEGM NAME=SEG4 ,PARENT=((SEG3,) ,(SEG1,PHYSICAL,DBD1))[1] ,BYTES=,FREQ= ,POINTER=(LPARNT,PAIRED)[1] ,RULES=

Note:

1. Specify symbolic or direct logical parent pointer. The direct-access pointer can be specified only when the logical parent is in an HDAM, HIDAM, PHDAM, or PHIDAM database.

Table 13. Statements that are required to define virtually paired bidirectional logical relationship

Statements for DBD1	Statements for DBD2
SEGM NAME=SEG1,PARENT= ,BYTES=,FREQ= ,POINTER=,RULES=	SEGM NAME=SEG3,PARENT= ,BYTES=,FREQ= ,POINTER=,RULES= SEGM NAME=SEG4 ,PARENT=SEG3 ,POINTER=PAIRED ,SOURCE=((SEG2,DATA,DBD1))

Table 13. Statements that are required to define virtually paired bidirectional logical relationship (continued)

Statements for DBD1	Statements for DBD2
SEGM NAME=SEG2 ,PARENT=((SEG1,) ,(SEG3,PHYSICAL,DBD2))[1] ,BYTES=,FREQ= ,POINTER=(LTWIN,LPARNT)[2] ,RULES=	LCHILD NAME=(SEG2,DBD1) ,POINTER=SNGL[3] ,PAIR=SEG4 ,RULES=[3]

Notes:

1. Specify symbolic or direct logical parent pointer. The direct-access pointer can be specified only when the logical parent is in an HDAM, HIDAM, PHDAM, or PHIDAM database.
2. Specify LTWIN or LTWINBWD for logical twin pointers.
3. Specify DNGL or DBLE for logical child pointers. The LCHILD RULES= parameter is used when either no sequence field or a nonunique sequence field has been defined for the virtual logical child or when the virtual logical child segment does not exist.

In the Virtually Paired Bidirectional Logical Relationship area of the following figure, a HISAM database can participate in a virtually paired logical relationship only when the real logical child is in an HDAM, HIDAM, PHDAM, or PHIDAM database and its logical parent is in the HISAM database.

The following figure illustrates how logical relationships and logical databases are defined. Part 1 depicts the physical data structures of a payroll database and a skills inventory database. Part 2 depicts the logical relationship between the physical data structures, NAMEMAST (in the Payroll database) and SKILNAME (in the Skills inventory database). Part 3 depicts the logical databases (SKILL and NAME) that can be defined as a result of the logical relationships. The new databases contain segments from both the NAMEMAST structure and the SKILNAME structure. Examples of DBD generation statements follow the figure.

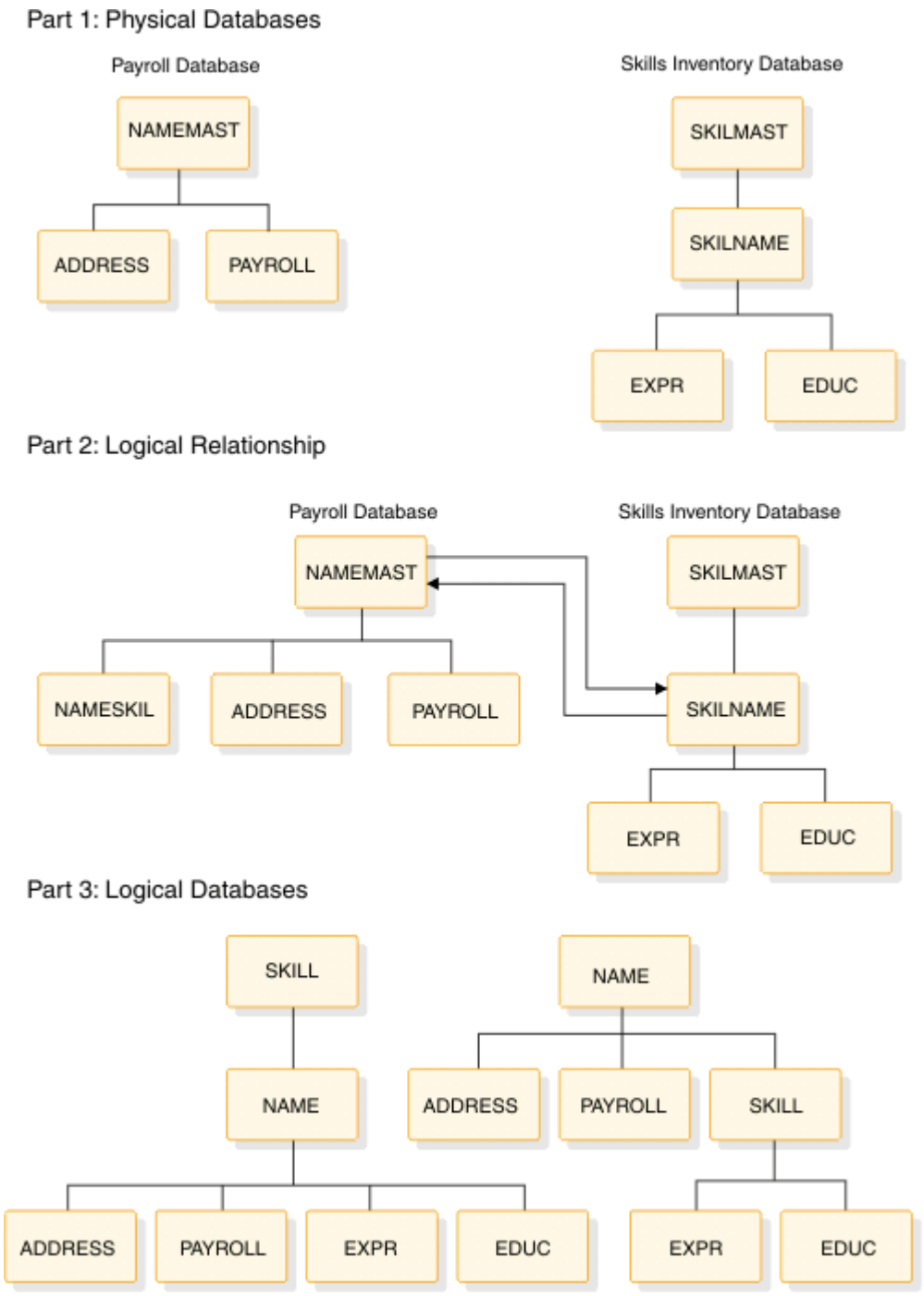


Figure 6. Logical relationship between physical databases and the resulting logical databases that can be defined

DBD generation statements examples

The following example shows the DBD generation statements necessary to define:

- The payroll and skills inventory data structures depicted in Part 2 of the preceding figure as a HIDAM and HDAM database with a virtually paired bidirectional logical relationship between the two databases
- The logical data structures depicted in Part 3 of the preceding figure as logical databases

```
DBD     NAME=PAYROLDB,ACCESS=HIDAM
DATASET DD1=PAYHIDAM,BLOCK=4096,SCAN=0
SEGM    NAME=NAMEMAST,PTR=TWINBWD,RULES=(VVV),                X
        BYTES=150
```

```
LCHILD NAME=(INDEX,INDEXDB),PTR=INDX
LCHILD  NAME=(SKILNAME,SKILLINV),PAIR=NAMESKIL,PTR=DBLE
FIELD NAME=(EMPLOYEE,SEQ,U),BYTES=60,START=1,TYPE=C
FIELD NAME=MANNBR,BYTES=15,START=61,TYPE=C
FIELD NAME=ADDR,BYTES=75,START=76,TYPE=C
SEGM NAME=NAMESKIL,PARENT=NAMEMAST,PTR=PAIRED,                  X
      SOURCE=((SKILNAME,DATA,SKILLINV))
FIELD NAME=(TYPE,SEQ,U),BYTES=21,START=1,TYPE=C
FIELD NAME=STDLEVL,BYTES=20,START=22,TYPE=C
SEGM   NAME=ADDRESS,BYTES=200,PARENT=NAMEMAST
FIELD NAME=(HOMEADDR,SEQ,U),BYTES=100,START=1,TYPE=C
FIELD NAME=COMAILOC,BYTES=100,START=101,TYPE=C
SEGM   NAME=PAYROLL,BYTES=100,PARENT=NAMEMAST
FIELD NAME=(BASICPAY,SEQ,U),BYTES=15,START=1,TYPE=P
FIELD NAME=HOURS,BYTES=15,START=51,TYPE=P
DBDGEN
FINISH
END

DBD    NAME=SKILLINV,ACCESS=HDAM,RMNAME=(DFSHDC40,20,500,824)
DATASET    DD1=SKILHDAM,BLOCK=4096,SCAN=0
SEGM  NAME=SKILMAST,BYTES=31,PTR=TWINBWD
FIELD NAME=(TYPE,SEQ,U),BYTES=21,START=1,TYPE=C
FIELD NAME=STDCODE,BYTES=10,START=22,TYPE=C
SEGM  NAME=SKILNAME,                                            X
      PARENT=((SKILMAST,DBLE),(NAMEMAST,P,PAYROLDB)),           X
      BYTES=80,PTR=(LPARNT,LTWINBWD,TWINBWD),                   X
      RULES=(VVV)
FIELD NAME=(EMPLOYEE,SEQ,U),START=1,BYTES=60,TYPE=C
FIELD NAME=(STDLEVL),BYTES=20,START=61,TYPE=C
SEGM  NAME=EXPR,BYTES=20,PTR=T,                                 X
      PARENT=((SKILNAME,SNGL))
FIELD NAME=PREVJOB,BYTES=10,START=1,TYPE=C
FIELD NAME=CLASSIF,BYTES=10,START=11,TYPE=C
SEGM  NAME=EDUC,BYTES=75,PTR=T,                                 X
      PARENT=((SKILNAME,SNGL))
FIELD NAME=GRADLEVL,BYTES=10,START=1,TYPE=C
FIELD NAME=SCHOOL,BYTES=65,START=11,TYPE=C
DBDGEN
FINISH
END

DBD    NAME=LOGICDB,ACCESS=LOGICAL
DATASET LOGICAL
SEGM   NAME=SKILL,SOURCE=((SKILMAST,,SKILLINV))
SEGM NAME=NAME,PARENT=SKILL,                                    X
       SOURCE=((SKILNAME,,SKILLINV),(NAMEMAST,,PAYROLDB))
SEGM NAME=ADDRESS,PARENT=NAME,SOURCE=((ADDRESS,,PAYROLDB))
SEGM NAME=PAYROLL,PARENT=NAME,SOURCE=((PAYROLL,,PAYROLDB))
SEGM NAME=EXPR,PARENT=NAME,SOURCE=((EXPR,,SKILLINV))
SEGM NAME=EDUC,PARENT=NAME,SOURCE=((EDUC,,SKILLINV))
DBDGEN
FINISH
END

BD    NAME=LOGIC1,ACCESS=LOGICAL
DATASET   LOGICAL
SEGM  NAME=NAME,SOURCE=((NAMEMAST,,PAYROLDB))
SEGM NAME=ADDRESS,PARENT=NAME,SOURCE=((ADDRESS,,PAYROLDB))
SEGM NAME=PAYROLL,PARENT=NAME,SOURCE=((PAYROLL,,PAYROLDB))
SEGM   NAME=SKILL,PARENT=NAME,                                  X
       SOURCE=((NAMESKIL,,PAYROLDB),(SKILMAST,,SKILLINV))
SEGM   NAME=EXPR,SOURCE=((EXPR,,SKILLINV)),PARENT=SKILL
SEGM   NAME=EDUC,SOURCE=((EDUC,,SKILLINV)),PARENT=SKILL
DBDGEN
FINISH
END
```

Related concepts

Creating logical relationships (Database Administration)

Examples with secondary indexes

These examples show the statements that are required to establish a secondary index relationship between a segment type in an indexed database and a segment type in a secondary index database.

The statements required when the index target and index source segment types are the same are shown in the following table.

Table 14. Same index source and target segment types

Indexed DBD	Index DBD
DBD NAME=DBD1,ACCESS= . . . SEGM NAME[1]=SEG1,PARENT= ,BYTES FIELD NAME=(FLD2,SEQ,...),BYTES= FIELD NAME=FLD1,BYTES= ,START LCHILD NAME=(SEG3,DBD2), POINTER[2]=INDX XDFLD NAME=XDNAME,SRCH=FLD1	DBD NAME=DBD2,ACCESS=INDEX . . . SEGM NAME=SEG3,PARENT=0,BYTES= FIELD NAME=(FLD2,SEQ,...),BYTES= ,START=1 LCHILD NAME=(SEG1,DBD1), INDEX=XDNAME,POINTER[2]=SNGL

Notes:

1. The index target segment type can be a root or a dependent segment type; it must not be either a logical child segment type or a dependent of a logical child segment type. The index source segment type must not be a logical child segment type.
2. The example is shown with direct pointers for the index pointer segment types in the index DBD. If symbolic pointing is desired, POINTER=SYMB should be specified on both LCHILD statements; symbolic pointing is required when the index target segment type is in a HISAM database.

In the following table, the index target and index source segment types are different. In both this table and the preceding one, only those operands pertinent to the secondary index relationships are shown.

Table 15. Different index source and target segment types

Indexed DBD	Index DBD
DBD NAME=DBD1,ACCESS= . . . SEGM NAME[1]=SEG1,BYTES=,PARENT= LCHILD NAME=(SEG4,DBD2), POINTER[2]=INDX XDFLD NAME=XDNAME,SEGMENT=SEG3, SRCH=FLD3,... SEGM NAME=SEG2,BYTES= ,PARENT=SEG1 SEGM NAME[1]=SEG3 ,PARENT=SEG2 FIELD NAME=FLD3,BYTES= ,START=	DBD NAME=DBD2,ACCESS=INDEX . . . SEGM NAME=SEG4,PARENT=0,BYTES= FIELD NAME=(FLD4,SEQ,...) ,START=1,BYTES= LCHILD NAME=(SEG1,DBD1), INDEX=XDNAME,POINTER[2]=SNGL

Notes:

1. The index target segment type can be a root or a dependent segment type. It must not be either a logical child segment type or a dependent of a logical child segment type. The index source segment type must not be a logical child segment type.
2. The example is shown with direct pointers for the index pointer segment types in the index DBD. If symbolic pointing is desired, POINTER=SYMB should be specified on both LCHILD statements; symbolic pointing is required when the index target segment type is in a HISAM database.

Example DBDs for full-function secondary index databases

The following figure shows a database, DTA1, that is indexed by two secondary index databases. The first secondary index, X1, uses the same segment for its index target segment and index source segment; the second secondary index, X2, has an index target segment that is different from its index source segment.

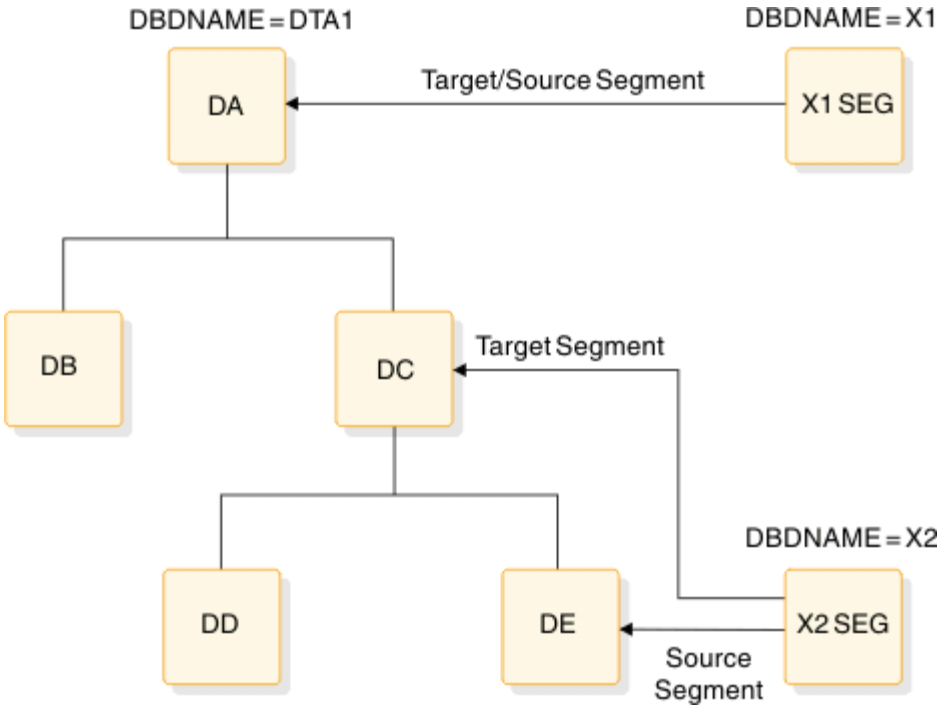


Figure 7. Database indexed by two secondary indexes

The following figure shows the DBD generation statement that defines the indexed database.

```
DBD     NAME=DTA1,ACCESS=HDAM,RMNAME=(DFSHDC40,20,500,824)
DATASET DD1=D1
SEGM    NAME=DA,PARENT=0,BYTES=15
FIELD   NAME=(DAF1,SEQ),BYTES=5,START=1
LCHILD  NAME=(X1SEG,X1),PTR=INDX
XDFLD   NAME=DAF1X,SRCH=DAF1
SEGM    NAME=DB,PARENT=DA,BYTES=20
FIELD   NAME=(DBF1,SEQ),BYTES=5,START=1
SEGM    NAME=DC,PARENT=DA,BYTES=20
FIELD   NAME=(DCF1,SEQ),BYTES=5,START=1
LCHILD  NAME=(X2SEG,X2),PTR=SYMB
XDFLD   NAME=DCF1X,SRCH=DEF1,SEGMENT=DE
SEGM    NAME=DD,PARENT=DC,BYTES=25
FIELD   NAME=(DDF1,SEQ),BYTES=5,START=1
SEGM    NAME=DE,PARENT=DC,BYTES=25
FIELD   NAME=(DEF1,SEQ),BYTES=5,START=1
DBDGEN
FINISH
END
```

The following figure shows the DBD generation statement that defines the secondary index database X1.

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

The following figure shows the DBD generation statement that defines the secondary index database X2.

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

Example DBDs for Fast Path secondary index databases

The following example shows a HISAM secondary index database DBD using unique key pointer segments.

```
DBDX   DBD      NAME=NAMESXDB,ACCESS=(INDEX,VSAM),FPINDEX=YES
       DATASET  DD1=NAMEKSDS
       SEGM     NAME=NAMEXSEG,PARENT=0,BYTES=15
       FIELD    NAME=(NAMESKEY,SEQ,U),BYTES=10,START=1
       LCHILD   NAME=(COURSE,EDUCDB),INDEX=NAMEINDX,PTR=SYMB
       DBDGEN
       FINISH
       END
```

The following example shows a HISAM secondary index database DBD using non-unique key pointer segments.

```
DBDX   DBD      NAME=NAMESXDB,ACCESS=(INDEX,VSAM),FPINDEX=YES
       DATASET  DD1=NAMEKSDS,OVFLW=NAMEESDS
       SEGM     NAME=NAMEXSEG,PARENT=0,BYTES=15
       FIELD    NAME=(NAMESKEY,SEQ,M),BYTES=10,START=1
       LCHILD   NAME=(COURSE,EDUCDB),INDEX=NAMEINDX,PTR=SYMB
       DBDGEN
       FINISH
       END
```

The following example shows a SHISAM secondary index database DBD using unique key pointer segments.

```
DBDX   DBD      NAME=NAMESXDB,ACCESS=(INDEX,SHISAM),FPINDEX=YES
       DATASET  DD1=NAMEKSDS
       SEGM     NAME=NAMEXSEG,PARENT=0,BYTES=15
       FIELD    NAME=(NAMESKEY,SEQ,U),BYTES=10,START=1
       LCHILD   NAME=(COURSE,EDUCDB),INDEX=NAMEINDX,PTR=SYMB
       DBDGEN
       FINISH
       END
```

The following three examples illustrate DEDB DBD definitions with secondary indexes, multiple secondary index segments, and a user partition defined.

The following example shows a primary DEDB database DBD with secondary indexing defined.

There are three secondary index databases defined for primary DEDB EDUCDB database: NAMESXDB, CLASSXDB, and INSTSXDB secondary index databases. The target segment for NAMESXDB secondary index is a root segment. The target segment, COURSE segment, for NAMESXDB is the same as the source segment. The target segments for CLASSXDB and INSTSXDB secondary indexes are not a root segment. The target segment, CLASS segment, for CLASSXDB is the same as the source segment. The target segment, INSTRUCT segment, for INSTSXDB is not the same as the source segment, COURSE segment.

```
DBD1   DBD      NAME=EDUCDB,ACCESS=DEDB,RMNAME=RMOD3
       AREA    NAME=EDAREA1,SIZE=1024,UOW=(100,10),ROOT=(236,36)

       SEGM    NAME=COURSE,PARENT=0,BYTES=100
       FIELD    NAME=(COURNO,SEQ,U),BYTES=5,START=1
       FIELD    NAME=COURNAME,BYTES=10,START=15

       LCHILD NAME=(NAMEXSEG,NAMESXDB),PTR=SYMB
```

```
        XDFLD   NAME=NAMEINDX,SRCH=COURNAME

        SEGM    NAME=CLASS,BYTES=50,PARENT=COURSE
        FIELD    NAME=(CLASSNO,SEQ,U),BYTES=4,START=7
        FIELD    NAME=CLASNAME,BYTES=10,START=15

        LCHILD NAME=(CLASXSEG,CLASSXDB),PTR=SYMB
        XDFLD  NAME=CLASINDX,SRCH=CLASNAME

          LCHILD NAME=(INSTXSEG,INSTSXDB),PTR=SYMB
        XDFLD  NAME=INSTINDX,SEGMENT=INSTRUCT,SRCH=INSTNAME

        SEGM    NAME=INSTRUCT,BYTES=50,PARENT=CLASS
        FIELD    NAME=(INSTNO,SEQ,U),BYTES=6,START=1
        FIELD    NAME=INSTPHNO,BYTES=10,START=11
        FIELD    NAME=INSTNAME,BYTES=20,START=21



        SEGM    NAME=STUDENT,BYTES=50,PARENT=CLASS
        FIELD    NAME=(STUDNO,SEQ,U),BYTES=6,START=1
        FIELD    NAME=STUDPHNO,BYTES=10,START=11
        FIELD    NAME=STUDNAME,BYTES=20,START=21
        FIELD    NAME=ENRLDATE,BYTES=6,START=41

        DBDGEN
        FINISH
        END
```

The following example shows a DEDB database DBD with multiple secondary index segments defined. The LCHILD statements define the same secondary index segment name (NAMEXSEG segment in the secondary database) and the same secondary index database name (NAMESXDB database). The XDFLD statements define the same sequence field name (NAMEINDX) for the secondary index segments with the different search fields (COURNAME and COURSECT) from the same source segment (COURSE segment).

The target segment of COURSE can be located using either a secondary index of course name (COURNAME) or a secondary index of course section number (COURSECT).

The search key lengths of the multiple secondary index segments (COURNAME and COURSECT) must be identical. In this example, they are both 10 bytes.

```
DBD1    DBD     NAME=EDUCMDB,ACCESS=DEDB,RMNAME=RMOD3
        AREA    NAME=EDMAREA1,SIZE=1024,UOW=(100,10),ROOT=(236,36)

        SEGM    NAME=COURSE,PARENT=0,BYTES=100
        FIELD   NAME=(COURNO,SEQ,U),BYTES=5,START=1
        FIELD   NAME=COURNAME,BYTES=10,START=15
        FIELD   NAME=COURSECT,BYTES=10,START=25

        LCHILD  NAME=(NAMEXSEG,NAMESXDB),PTR=SYMB,MULTISEG=YES
        XDFLD   NAME=NAMEINDX,SRCH=COURNAME
        LCHILD  NAME=(NAMEXSEG,NAMESXDB),PTR=SYMB,MULTISEG=YES
        XDFLD   NAME=NAMEINDX,SRCH=COURSECT

        SEGM    NAME=CLASS,BYTES=50,PARENT=COURSE
        FIELD   NAME=(CLASSNO,SEQ,U),BYTES=4,START=7
        FIELD   NAME=CLASNAME,BYTES=10,START=15

        SEGM    NAME=INSTRUCT,BYTES=50,PARENT=CLASS
        FIELD   NAME=(INSTNO,SEQ,U),BYTES=6,START=1
        FIELD   NAME=INSTPHNO,BYTES=10,START=11
        FIELD   NAME=INSTNAME,BYTES=20,START=21

        SEGM    NAME=STUDENT,BYTES=50,PARENT=CLASS
        FIELD   NAME=(STUDNO,SEQ,U),BYTES=6,START=1
        FIELD   NAME=STUDPHNO,BYTES=10,START=11
        FIELD   NAME=STUDNAME,BYTES=20,START=21
        FIELD   NAME=ENRLDATE,BYTES=6,START=41
        DBDGEN
        FINISH
        END
```

The following example shows a primary DEDB database DBD with secondary indexing defined using user partitioning for a HISAM secondary index database or a SHISAM secondary index database. There are two user partitions specified on the LCHILD statement: NAMSXDB1 and NAMSXDB2.

PSELRTN=DBFPSE00 is the user partition selection exit. The user partition selection option is PSELOPT=SNGL which indicates only the selected user partition database is used to access the primary DEDB database in the user partition group. When subsequent qualified GN calls with no SSA using the PCB with the PROCSEQD= parameter are issued, a GB status code is returned to an application to indicate end of database after the last pointer segment in the selected user partition is used to access the segment in the primary DEDB database.

```
DBD1   DBD    NAME=EDUCUDB,ACCESS=DEDB,RMNAME=RMOD3
       AREA   NAME=EDUAREA1,SIZE=1024,UOW=(100,10),ROOT=(236,36)

       SEGM   NAME=COURSE,PARENT=0,BYTES=100
       FIELD  NAME=(COURNO,SEQ,U),BYTES=5,START=1
       FIELD  NAME=COURNAME,BYTES=10,START=15

       LCHILD NAME=(NAMEXSEG,(NAMESXB1,NAMSXDB2)),PTR=SYMB
       XDFLD  NAME=XNAME,SRCH=COURNAME,PSELRTN=DBFPSE00,PSELOPT=SNGL

       SEGM   NAME=CLASS,BYTES=50,PARENT=COURSE
       FIELD  NAME=(CLASSNO,SEQ,U),BYTES=4,START=7
       FIELD  NAME=CLASNAME,BYTES=10,START=15

       SEGM   NAME=INSTRUCT,BYTES=50,PARENT=CLASS
       FIELD  NAME=(INSTNO,SEQ,U),BYTES=6,START=1
       FIELD  NAME=INSTPHNO,BYTES=10,START=11
       FIELD  NAME=INSTNAME,BYTES=20,START=21

       SEGM   NAME=STUDENT,BYTES=50,PARENT=CLASS
       FIELD  NAME=(STUDNO,SEQ,U),BYTES=6,START=1
       FIELD  NAME=STUDPHNO,BYTES=10,START=11
       FIELD  NAME=STUDNAME,BYTES=20,START=21
       FIELD  NAME=ENRLDATE,BYTES=6,START=41

       DBDGEN
       FINISH
       END
```

Related concepts

Creating secondary indexes (Database Administration)

Running the DBDGEN procedure

Running the DBDGEN procedure is a two step assemble and bind procedure that produces database definition blocks. The procedure is built during SMP/E processing and named DFSDBDGN in the ADFSPROC and SDFSPROC data sets. Use the DFSPROCB JCL to rename DFSDBDGN to DBDGEN.

Recommendations

Currently, no recommendations are documented for the DBD Generation utility.

JCL for the DBDGEN utility

The following example shows the JCL for the DBDGEN utility.

```
//       PROC MBR=TEMPNAME,SOUT=A,RGN=0M,SYS2=,
//               NODE1=IMS,
//               NODE2=IMS
//C      EXEC PGM=ASMA90,REGION=&RGN,
//            PARM=(OBJECT,NODECK,NODBCS,
//            'SIZE(MAX,ABOVE)')
//SYSLIB   DD DSN=&NODE2..&SYS2.SDFSMAC,DISP=SHR
//SYSLIN   DD UNIT=SYSDA,DISP=(,PASS),
//         SPACE=(80,(100,100),RLSE),
//         DCB=(BLKSIZE=80,RECFM=F,LRECL=80)
//SYSPRINT DD SYSOUT=&SOUT,DCB=BLKSIZE=1089,
//         SPACE=(121,(300,300),RLSE,,ROUND)
//SYSUT1   DD UNIT=SYSDA,DISP=(,DELETE),
//         SPACE=(CYL,(10,5))
//L      EXEC PGM=IEWL,PARM='XREF,LIST',
//            COND=(0,LT,C),REGION=4M
//SYSLIN   DD DSN=*.C.SYSLIN,DISP=(OLD,DELETE)
```

```
//SYSPRINT DD SYSOUT=&SOUT,DCB=BLKSIZE=1089,
//         SPACE=(121,(90,90),RLSE)
//SYSLMOD  DD DISP=SHR,
//         DSN=&NODE1..&SYS2.DBDLIB(&MBR)
//SYSUT1   DD UNIT=(SYSDA,SEP=(SYSLMOD,SYSLIN)),
//         SPACE=(1024,(100,10),RLSE),DISP=(,DELETE)
```

Procedure to invoke the DBDGEN

To process a request for a DBDGEN, the DBD generation control statements must be created and appended to the JCL (shown in the following figure) which invokes the DBDGEN procedure.

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

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

JCL parameters

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

RGN=
Specifies the region size for this execution. The default is 256 KB.

SOUT=
Specifies the class assigned to SYSOUT DD statements.

SYS2=
Specifies an optional second level dsname qualifier for those data sets which are designated as "Optional Replicate" in an XRF complex. When specified, the parameter must be enclosed in quotes and must include a trailing period; for example, SYS2='IMSA.'.

Step C

Step C is the assembly step. The following DD statement is needed for this step.

SYSIN DD
Defines the input data sets to step C. These DD statements must be provided when invoking the procedure.

Step L

Step L is the bind step.

Example: This step can be run using AMODE=31, RMODE=24 instead of the default AMODE=24, RMODE=24 by adding AMODE=31 to the bind EXEC statement PARM list as shown as follows:

```
//L      EXEC  PGM=IEWL,PARM='XREF,LIST,AMODE=31',
//             COND=(0,LT,C),REGION=120K
```

If you do not specify different values for AMODE or RMODE, the default values are in effect. You must always run the bind step with RMODE=24. The following DD statement is needed for this step.

IMS.DBDLIB DD
Defines an output partitioned data set, IMS.DBDLIB, for the binder.

Chapter 3. MFS Device Characteristics Table utility (DFSUTB00)

Use the Message Format Service Device Characteristics Table (MFSDCT) utility (DFSUTB00) to define new screen sizes in a descriptor member of the IMS.PROCLIB library without performing an IMS system definition. These new screen size definitions are added to the screen sizes that were previously defined.

The MFSDCT (DFSUTB00) utility procedure consists of the following steps:

1. The DFSUTB00 program is executed to initiate several functions. The DFSUTB00 program:
   - Reads one or two descriptor members from PROCLIB and uses only the new device descriptors as input.
   - Builds DCTENTRY statements for each device descriptor.
   - Optionally loads an existing device characteristics table from JOBLIB/STEPLIB data sets (usually from the IMS.SDFSRESL library) and then builds DCENTRY statements for each DCT entry.
   - Invokes the assembler, passing the DCTENTRY statements and the DCTBLD and MFSINIT macros as input.
   - Readies the output from the assembler as an updated or new device characteristics table and as a new set of default MFS format definitions. (This output is split into separate files for later processing.)
2. The assembler is invoked to assemble the new device characteristics table.
3. The binder is invoked to bind the new device characteristics table into the IMS.SDFSRESL.
4. Phase 1 of the MFS Language utility generates new default MFS format control blocks.
5. Phase 2 of the MFS Language utility puts the new default MFS format control blocks into the IMS.FORMAT library.

Subsections:

- “Restrictions” on page 161
- “Prerequisites” on page 161
- “Requirements” on page 161
- “Recommendations” on page 162
- “JCL specifications” on page 162
- “Return codes” on page 163

Restrictions

The following restrictions apply to this utility:

- The utility ignores all other descriptors while reading the one or two descriptor members from PROCLIB.
- At least one device descriptor must be specified or the utility terminates.

Prerequisites

Currently, no prerequisites are documented for the DFSUTB00 utility.

Requirements

To run the DFSUTB00 utility you must satisfy MFS device description format requirements.

© Copyright IBM Corp. 1974, 2024

MFS device descriptors are used by the MFS Device Characteristics Table utility to update screen size in the DCT and generate new MFS default formats without system definition.

Recommendations

Currently, no recommendations are documented for the DFSUTB00 utility.

JCL specifications

The MFSDCT procedure requires the procedure statement, the EXEC statement, DD statements, and MFS device descriptions.

Procedure statement

The procedure statement must be in the form shown in the following example.

```
PROC RGN=4M,SOUT=A,SYS2=,PXREF=NOXREF,
PCOMP=NOCOMP,PSUBS=NOSUBS,PDIAG=NODIAG,
COMPR=NOCOMPRESS,COMPR2=COMPRESS,
LN=55,SN=8,DEVCHAR=0,COMPR3=NOCOMPREND,
DIRUPDT=UPDATE,DCTSUF=,
DSCTSUF=,DSCMSUF=,FMTMAST=N
```

In addition to the optional keyword parameters, you might need to specify the following parameters depending on the other parameters you specify. (*x* is the alphanumeric suffix character that you are appending to the member name.)

DCTSUF=*x*
Specifies the suffix character to be appended to DFSUDT0. The name DFSUDT0*x* identifies the device characteristics table to which new definitions are added. This suffix character corresponds to the value specified in the SUFFIX= keyword of the IMSGEN macro. If a suffix character is not specified, a completely new device characteristics table is built from just the device descriptors.

DSCTSUF=*x*
Specifies the suffix character to be appended to DFSDSCT. The name DFSDSCT*x* identifies a descriptor member. This suffix character corresponds to the value specified in the IMS procedure DSCT= keyword. This parameter is required if DSCMSUF= is not specified.

DSCMSUF=*x*
Specifies the suffix character to be appended to DFSDSCM. The name DFSDSCM*x* identifies a descriptor member. This suffix character corresponds to the value specified in the SUFFIX= keyword of the IMSGEN macro. This parameter is required if DSCTSUF= is not specified.

FMTMAST=Y/N
Specifies whether (Y) or not (N) the IMS-provided support for MFS is to be used on the master terminal.

EXEC statement

The EXEC statement determines that a device characteristics table is created. It also specifies the name for the desired descriptor member and the name of the updated or new device characteristics table. Each of the five steps in this procedure names a different program for execution.

The following figure shows the five steps of the MFSDCT (DFSUTB00) utility.

```
//S1      EXEC PGM=DFSUTB00,REGION=&RGN,
//        PARM=('DCTSUF=&DCTSUF,DSCTSUF=&DSCTSUF'
//        'DSCMSUF=&DSCMSUF,DEVCHAR=&DEVCHAR')
//S2      EXEC PGM=ASMA90,REGION=&RGN,
//        PARM=('OBJECT,NODECK,NOLIST',
//        COND=(0,LT)'
```

```
//S3      EXEC PGM=IEWL,
//        PARM=('SIZE=880K,64K),NCAL,LET,REUS,XREF,LIST',
//        REGION=&RGN,
//        COND=(0,LT)
//S4      EXEC PGM=DFSUPAA0,REGION=&RGN,
//        PARM=(&PXREF,&PCOMP,&PSUBS,&PDIAG,&COMPR,;
//        'LINECNT=&LN,STOPRC=&SN,DEVCHAR=&DEVCHAR'),
//        COND=(0,LT)
//S5      EXEC PGM=DFSUNUB0,REGION=&RGN,
//        PARM=(&COMPR2,&COMPR3,&DIRUPDT,;
//        'DEVCHAR=&DEVCHAR'),COND=((0,LT,S1),
//        (0,LT,S2),(0,LT,S3),(8,LT,S4))
```

DD statements

The following ddnames are used in step 1 of the MFSDCT procedure.

DCT
Defines the temporary data set for the updated or new device characteristics table as output from the assembler with the ddname SYSLIN (step 2) and as input to the binder with ddname DCT (step 3).

DCTIN
Defines a temporary data set for the device characteristics table as input to the assembler (step 2).

DCTLNK
Defines the temporary data set for the bind control statements for step 3.

DEFLTS
Defines the temporary data set for the default MFS format definitions for MFS Language utility input (step 4).

PROCLIB
Defines the libraries containing the descriptor members DFSDSCM*x* and DFSDSCT*x*.

STEPLIB
Defines the libraries containing the program DFSUTB00 and the device characteristics table specified in the DCTSUF= parameter.

SYSIN
Defines the temporary file containing the generated DCENTRY statements.

SYSLIN
Defines the temporary data set for the updated or new device characteristics table as output from the assembler (step 2) and as input to the binder with ddname DCT (step 3).

SYSLIB
Defines the libraries containing IMS and z/OS macros.

SYSPRINT
Defines the data set for all of the printed output from step 1, including error messages and output from steps 2 and 3.

SYSPUNCH
Defines the temporary file containing the object module output from the assembler. The output is the device characteristics table, followed immediately by the default MFS format definitions.

SYSUT1
Defines an assembler and binder work data set.

SYSLMOD
Defines the IMS.SDFSRESL data set to contain the new or modified device characteristics table.

Return codes

Return codes are based on the error message.

Running the DFSUTB00 utility

You can invoke the MFS Device Characteristics Table utility by running the Message Format Service Device Characteristics Table (MFSDCT) procedure.

Parameters

You can specify the following execution parameters in the MFSDCT procedure:

- NODE1
- NODE2
- NODE3: specifies the high-level qualifier of the IMS data sets, which corresponds to the third node name specified in the **NODE** parameter of the IMSGEN macro.

NODE1 and NODE2 are described in Parameter descriptions for IMS procedures (System Definition)

Message Format Service Device Characteristics Table procedure

```
//         PROC RGN=4M,SOUT=A,SYS2=,PXREF=NOXREF,
//            PCOMP=NOCOMP,PSUBS=NOSUBS,PDIAG=NODIAG,
//            COMPR=NOCOMPRESS,COMPR2=COMPRESS,
//            LN=55,SN=8,DEVCHAR=0,COMPR3=NOCOMPREND,
//            DIRUPDT=UPDATE,DCTSUF=,
//            DSCTSUF=,DSCMSUF=,FMTMAST=N,
//            NODE1=IMS,
//            NODE2=IMS,
//            NODE3=IMS
//S1       EXEC PGM=DFSUTB00,REGION=&RGN,
//         PARM=('DCTSUF=&DCTSUF,DSCTSUF=&DSCTSUF',
//         'DSCMSUF=&DSCMSUF,DEVCHAR=&DEVCHAR'
//         'FMTMAST=&FMTMAST')
//STEPLIB  DD DSN=&NODE2..&SYS2.SDFSRESL,DISP=SHR
//SYSLIB   DD DSN=&NODE3..ADFSMAC,DISP=SHR
//         DD DSN=SYS1.MACLIB,DISP=SHR
//         DD DSN=SYS1.MODGEN,DISP=SHR
//         DD DSN=SYS1.SDFSMAC,DISP=SHR
//PROCLIB  DD DSN=&NODE2..&SYS2.PROCLIB,DISP=SHR
//SYSIN    DD DSN=&&SYSIN,UNIT=SYSDA,
//         SPACE=(CYL,(1,1)),DCB=BLKSIZE=800
//SYSPUNCH DD DSN=&&SYSPUNCH,UNIT=SYSDA,
//         SPACE=(CYL,(1,1)),DCB=BLKSIZE=800
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSPRINT DD SYSOUT=&SOUT,
//         DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYSUDUMP DD SYSOUT=&SOUT
//DCTIN    DD DSN=&&DCTIN,DISP=(NEW,PASS),
//         UNIT=SYSDA,SPACE=(CYL,(1,1))
//DEFLTS   DD DSN=&&DEFLTS,DISP=(NEW,PASS),
//         UNIT=SYSDA,SPACE=(CYL,(1,1))
//DCTLNK   DD DSN=&&DCTLNK,DISP=(NEW,PASS),
//         UNIT=SYSDA,SPACE=(TRK,(1,1)),
//         DCB=BLKSIZE=800
//S2       EXEC PGM=ASMA90,REGION=& RGN,
//            PARM='OBJECT,NODECK,NOLIST',
//            COND=(0,LT)
//SYSLIB   DD DSN=&NODE3..ADFSMAC,DISP=SHR
//         DD DSN=SYS1.MACLIB,DISP=SHR
//         DD DSN=SYS1.MODGEN,DISP=SHR
//SYSLIN   DD DSN=&&DCT,DISP=(NEW,PASS)
//         UNIT=SYSDA,SPACE=(CYL,(1,1)),
//         DCB=BLKSIZE=800<
//SYSPRINT DD SYSOUT=&SOUT,
//         DCB=(BLKSIZE=605),
//         SPACE=(605,(100,50),RLSE,,ROUND)
//SYSUT1   DD UNIT=SYSDA,DISP=(,DELETE),
//         SPACE=(CYL,(15,15))
//SYSIN    DD DSN=&&DCTIN,DISP=(OLD,DELETE)
//S3       EXEC PGM=IEWL,
//            PARM=('SIZE=(880K,64K)',NCAL,LET,REUS,
//            XREF,LIST),
//            REGION=&RGN,
//            COND=(0,LT)
//SYSPRINT DD SYSOUT=&SOUT,
//         DCB=(RECFM=FBA,LRECL=121,BLKSIZE=605),
```

```
//         SPACE=(605,(10,10),RLSE,,ROUND)
//SYSLMOD  DD DSN=&NODE2..&SYS2.SDFSRESL,DISP=SHR
//SYSUT1   DD UNIT=(SYSDA,SEP=(SYSLMOD,SYSPUNCH)),
//         SPACE=(CYL,(10,1))
//SYSLIN   DD DSN=&&DCTLNK,DISP=(SHR,DELETE)
//DCT      DD DSN=&&DCT,DISP=(SHR,DELETE)
//S4       EXEC PGM=DFSUPAA0,REGION=&RGN,
//            PARM=(&PXREF,&PCOMP,&PSUBS,&PDIAG,;
//            &COMPR,'LINECNT=&LN,STOPRC=&SN',
//            'DEVCHAR=&DEVCHAR'),COND=(0,LT)
//STEPLIB  DD DSN=&NODE2..&SYS2.SDFSRESL,DISP=SHR
//*SYSLIB - USER OPTION
//SYSIN    DD DSN=&&DEFLTS,DISP=(OLD,DELETE)
//REFIN    DD DSN=&NODE1..&SYS2.REFERAL,DISP=OLD
//REFOUT   DD DSN=&NODE1..&SYS2.REFERAL,DISP=OLD
//REFRD    DD DSN=&NODE1..&SYS2.REFERAL,DISP=OLD
//SYSTEXT  DD DSN=&&TXTPASS,UNIT=SYSDA,
//         SPACE=(CYL,(1,1)),DCB=BLKSIZE=800
//SYSUT3   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//DUMMY    DD DISP=SHR,
//         DSN=&NODE2..&SYS2.PROCLIB(REFCPY)
//UTPRINT  DD SYSOUT=&SOUT
//SYSPRINT DD SYSOUT=&SOUT,
//         DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYSUDUMP DD SYSOUT=&SOUT
//SEQBLKS  DD DSN=&&BLKS,DISP=(NEW,PASS),
//         UNIT=SYSDA,SPACE=(CYL,(1,1))
//S5       EXEC PGM=DFSUNUB0,REGION=&RGN,
//            PARM=(&COMPR2,&COMPR3,&DIRUPDT,;
//            'DEVCHAR=&DEVCHAR'),COND=((0,LT,S1),
//            (0,LT,S2),(0,LT,S3),(8,LT,S4))
//STEPLIB  DD DSN=&NODE2..&SYS2.SDFSRESL,DISP=SHR
//SEQBLKS  DD DSN=&&BLKS,DISP=(OLD,DELETE)
//UTPRINT  DD SYSOUT=&SOUT,
//         DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYSUDUMP DD SYSOUT=&SOUT
//FORMAT   DD DSN=&NODE1..&SYS2.FORMAT,DISP=SHR
//DUMMY    DD DISP=SHR,
//         DSN=&NODE2..&SYS2.PROCLIB(FMTCPY)
//SYSPRINT DD SYSOUT=&SOUT
//SYSUT3   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
```

Chapter 4. MFS Language utility (DFSUPAA0)

Use the MFS Language utility (DFSUPAA0) to create and store the Message Format Service (MFS) control blocks.

The intermediate text block (ITB) form of the control blocks is placed in the IMS.REFERAL library. The control blocks are placed in the IMS.FORMAT library for use during normal IMS operation.

Definition: One format and all the messages that refer to it in their SOR= operand make up a *format set*.

The MFS Language utility has three modes of operation: standard, batch, and test. In all three modes, the utility is executed offline, accepts the same control statements, and produces the same kinds of ITBs and control blocks. The modes differ in their use of the MFS libraries. Accordingly, they use different procedures.

In standard mode, ITBs written in IMS.REFERAL are converted to control blocks and placed in the staging library, IMS.FORMAT, by the MFSUTL procedure. Because the control blocks are placed in the staging library and not the active library, the standard mode can run concurrently with the online IMS control region.

Batch mode differs from standard mode, in that the MFSBTCH1 procedure places the created control blocks in a special library, IMS.MFSBATCH, for later transfer by the MFSBTCH2 procedure (in another job) to the staging library, IMS.FORMAT.

In test mode, the MFSTEST procedure creates control blocks and places them in a separate IMS.TFORMAT library. The control blocks can be tested without interfering with online operation and can operate concurrently with the online IMS control region.

SMP/E processing generates the following procedures:

MFSUTL
A two-step standard mode execution procedure of the MFS Language utility for creating MFS online control blocks and placing these blocks into the IMS.FORMAT library.

MFSBTCH1
A one-step batch mode execution procedure of the MFS Language utility for creating and accumulating MFS online blocks.

MFSBTCH2
A one-step batch mode execution procedure of the MFS Language utility for placing the accumulated MFS online control blocks (from MFSBTCH1) into the IMS.FORMAT library.

MFSBACK
A two-step execution procedure to back up the MFS libraries. If the optional MFSTEST facility is used, MFSBACK contains an additional step to back up the test library.

MFSREST
A two-step execution procedure to restore the MFS libraries. If the optional MFSTEST facility is used, MFSREST contains an additional step to restore the test library.

MFSRVC
A one-step execution procedure for maintaining the MFS libraries.

If MFSTEST mode is selected during system definition, an additional procedure is generated:

MFSTEST
A two-step test mode execution procedure of the MFS Language utility for creating MFS online blocks and placing them into the IMS.TFORMAT library.

In addition to the procedures for creating new or replacement control blocks, the MFS Language utility includes MFSBACK and MFSREST procedures for backup and restore operations in MFS libraries.

Delete and listing operations are performed by the service utility.

Subsections:

© Copyright IBM Corp. 1974, 2024

- “Restrictions” on page 168
- “Prerequisites” on page 168
- “Requirements” on page 168
- “Recommendations” on page 168
- “JCL specifications” on page 168

Restrictions

Do not execute the MFSTEST procedure concurrently with itself or any other program or procedure that utilizes the MFS libraries. To test the control blocks in IMS.TFORMAT, the terminal operator enters the **/ TEST MFS** command. Then, test control blocks from IMS.TFORMAT (as well as online control blocks from the active format library, if necessary) are read into a buffer for test operation. After successful testing, the control blocks can be placed in the staging IMS.FORMAT library by recompiling the source statements using the MFSUTL procedure.

Prerequisites

Currently, no prerequisites are documented for the DFSUPAA0 utility.

Requirements

Currently, no requirements are documented for the DFSUPAA0 utility.

Recommendations

Currently, no recommendations are documented for the DFSUPAA0 utility.

JCL specifications

The DFSUPAA0 utility requires an EXEC statement and DD statements for the MFSUTL, MFSBTCH1, and MFSTEST procedures.

When Step 1 (S1) executes (in the MFSUTL, MFSBTCH1, and MFSTEST procedures), the following parameters can be specified in the PARM keyword of the EXEC statement.

PXREF= NOXREF | XREF
: Specifies whether (XREF) or not (NOXREF) a sorted cross reference listing should be provided. The default value is NOXREF. A sorted cross reference listing includes a list of all labels and related references.

PCOMP= NOCOMP | COMP
: Specifies whether (COMP or COMPOSITE) or not (NOCOMP) the composite or final version of the statement, after error recovery or substitution has modified it, is printed. The default value is NOCOMP. The composite statement reflects syntactic assumptions made during error recovery. Semantic assumptions do not appear in the composite statement but are reflected in the intermediate text blocks. If the repetitive generation function for MFLD/DFLD statements is used, COMP also causes the generated statements to be printed; NOCOMP suppresses this printing.

PSUBS= NOSUBS | SUBS
: Specifies whether (SUBS or SUBSTITUTE) or not (NOSUBS) the substitution variable and its equated value are printed when the substitution variable is encountered in the operand field of a statement. The default value is NOSUBS.

PDIAG= NODIAG | DIAG
Specifies whether (DIAG or DIAGNOSTIC) or not (NODIAG) the XREF, COMP, and SUBS options should all be set on. In addition, diagnostic information is printed. The default value is NODIAG, which has no effect on the XREF, COMP, and SUBS options but suppresses printing of the diagnostic information.

COMPR= NOCOMPRESS | COMPRESS
Specifies whether (COMPRESS) or not (NOCOMPRESS) the IMS.REFERAL library is to be compressed before new ITBs are added. The default value is NOCOMPRESS.

LN= 55 | *nn*
Specifies how many lines per page should be printed. The default value is 55.

SN= 08 | *nn*
Specifies the severity code compare value. MSG, FMT, and TABLE blocks whose error severity equals or exceeds this value are not written to the IMS.REFERAL library. The default value is 08.

DEVCHAR= 0 | *x*
Specifies the alphanumeric suffix character (x) to be appended to DFSUDT0. The name DFSUDT0 identifies the desired device characteristics table. This suffix character (x) corresponds to the value specified in the SUFFIX= keyword of the IMSGEN macro. The default is zero (0).

In the execution of the MFSRVC procedure, one parameter can be specified. The DEVCHAR=0 or x parameter specifies the alphanumeric suffix character (x) to be used for the device characteristics table, when no suffix is specified in the LIST control statement parameter DEVCHAR. The default is zero.

In the execution of Step 2 (S2) in the MFSUTL and MFSBTCH2 procedures, three parameters can be specified in the EXEC statement's PARM keyword:

COMPR2= COMPRESS | NOCOMPRESS
Specifies whether (COMPRESS) or not (NOCOMPRESS) the IMS.FORMAT library is to be compressed before new control blocks are added. The default value is COMPRESS.

COMPR3= COMPREND | NOCOMPREND
Specifies whether (COMPREND) or not (NOCOMPREND) the data set with the ddname of FORMAT is compressed after all format blocks have been added/replaced and the index directory ($$IMSDIR) has been updated.

DIRUPDT= UPDATE | NOUPDATE
Specifies whether (UPDATE) or not (NOUPDATE) the special index directory ($$IMSDIR) is automatically updated after a block has been deleted from a format library. You can bypass the $$IMSDIR update by specifying NOUPDATE. The default is UPDATE.

In the execution of Step 2 (S2) in the MFSTEST procedure, the PARM='TEST' parameter must be specified.

Other EXEC statement parameters that can be specified are:

RGN=
Specifies the region size for this execution. The default is 360K.

SOUT=
Specifies the SYSOUT class. The default is A.

SNODE=
Specifies the node that can be assigned to the MFS utility data set name. The default value is IMS.

SOR=
Specifies the library name that can be assigned to the MFS utility library for SYSIN or SYSLIB. The default value is NOLIB.

MBR=
Specifies the member name that can be assigned to the MFS utility member for SYSIN. The default is NOMBR.

EXEC statement

EXEC statement parameters supported by the MFS Language utility have variable compilation control functions.

The format of the EXEC statement is:

```
//S1       EXEC PGM=DFSUPAA0,REGION=&RGN,
//             PARM=(&PXREF,&PCOMP,&PSUBS,&PDIAG,
//             &COMPR,'LINECNT=&LN,STOPRC=&SN',
//             'DEVCHAR=&DEVCHAR')
```

Parameters can be specified on the EXEC statement for the preprocessor and phase 1 to:

- Control the printed output
- Compress the reference library (IMS.REFERAL)
- Request diagnostic information
- Indicate which MFS device characteristics table is to be used
- Prevent control blocks with a specified level of error from being written in IMS.REFERAL

Parameters can also be specified on the EXEC statement for phase 2 to specify whether IMS.FORMAT and IMS.REFERAL should be compressed and whether $$IMSDIR should be automatically updated after deletions.

The DEVCHAR parameter specifies the suffix of the MFS device characteristics table to be used. The device characteristics table is accessed only if DEV TYPE=3270-An (where n is 1 to 15) is coded as input to the MFS Language utility.

The EXEC statement parameters supported by the MFS Language utility have variable compilation control functions. The parameters that can be specified are:

NOXREF | XREF
Specifies whether (XREF) or not (NOXREF) a sorted cross-reference listing should be provided. A sorted cross-reference listing includes a list of all the labels and related references. The default is NOXREF.

NOCOMP | COMP
Specifies whether (COMP or COMPOSITE) or not (NOCOMP) the composite or final version of the statement, after error recovery or substitution has modified it, will be printed. A composite statement reflects syntactic assumptions made during error recovery. Semantic assumptions do not appear in composite statements but are reflected in the intermediate text blocks. The default is NOCOMP.

NOSUBS | SUBS
Specifies whether (SUBS or SUBSTITUTE) or not (NOSUBS) any statement containing a substitution variable (EQU operand) is printed. The default is NOSUBS.

NODIAG | DIAG
Specifies whether (DIAG or DIAGNOSTIC) or not (NODIAG) the XREF, COMP, and SUBS options should be set on and diagnostic information be printed. The default is NODIAG, which has no effect on the setting of the XREF, COMP, and SUBS options but suppresses printing of the diagnostic information.

NOCOMPRESS | COMPRESS
Specifies whether (COMPRESS) or not (NOCOMPRESS) the IMS.REFERAL library is to be compressed before new ITBs are added. The default is NOCOMPRESS.

DIRUPDT= UPDATE | NOUPDATE
Specifies whether (UPDATE) or not (NOUPDATE) the special index directory ($$IMSDIR) will be automatically updated after one or more blocks have been deleted from a format library. You can bypass the $$IMSDIR update by specifying NOUPDATE. The default is UPDATE.

LINECNT=*nn*
Specifies how many lines per page should be printed. The default is 55.

STOPRC=*nn*
Specifies the severity code compare value. MSG, FMT, and TABLE blocks whose error severity equals or exceeds this value will not be written to the IMS.REFERAL library. The default is 08.

DEVCHAR=*n* | x
Specifies the alphanumeric suffix character (x) used as the final character of the name of the device characteristics table DFSUDT0x loaded when DEV TYPE=3270-An is encountered. The default is zero (DFSUDT00).

The definition statements are described in the sequence shown, with the DO and ENDDO compilation statements where they would normally be coded—before and after the MFLD or DFLD statements. The compilation statement formats are sequenced according to related function (if any)—ALPHA; COPY; EQU and RESCAN (equate processing); STACK and UNSTACK (stacking SYSIN/SYSLIB records); TITLE, PRINT, SPACE, and EJECT (SYSPRINT listing control); and END.

Estimating MFSUTL and MFSTEST region parameters

The following steps help you estimate the main storage requirements that you should specify in the RGN= parameter of the EXEC statement invoking the MFSUTL and MFSTEST procedures.

1. Calculate statement base count. For the input to the MFS Language utility, determine the largest (number of statements) device format to be processed and the largest message descriptor related to the format. Add the total number of statements contained in these two control blocks to obtain the statement base count.

   For the processing of a specific user-supplied MSG or FMT ITB, the utility reprocesses all related MSG or FMT ITBs saved from the IMS.REFERAL data set to ensure compatible linkage between all related online blocks. These reprocessed ITBs must be analyzed as well for the process of obtaining the statement base count.
2. Estimate Region Requirements. Multiply the statement base count by 214 and add 300000 to the result. Round the resulting value to the next highest multiple of 2048. The result is an estimate of the main storage requirements which should be specified in the RGN= parameter of the EXEC statement invoking the MFSUTL and MFSTEST procedures.

   Complex formats with a large number of literal DFLD statements in relation to the statement base count can exceed the estimate.

DD statements

The data set names used in the MFSUTL, MFSBTCH1, MFSBTCH2, and MFSTEST procedures fit installation needs. The ddnames used and the data sets they refer to are:

REFIN
REFOUT
REFRD
Refers to the MFS reference library, except when used in the MFSTEST procedure. In MFSTEST, REFIN and REFRD refer to the MFS reference library; REFOUT is a temporary data set.

FORMAT
Refers to the MFS control block library. In MFSTEST, this ddname refers to the MFS test control block library.

SYSLIB
Refers to an optional user library from which input can be copied.

SYSIN
Refers to the input data set, which can be a sequential data set or a member of a partitioned data set.

DUMMY
Refers to the IMS procedure library, which contains control statements used to compress the MFS reference and control block libraries.

SYSUT3
SYSUT4
Are ddnames for data sets used during the data set compression as work data sets.

DUMMY, SYSUT3, and SYSUT4 can all be omitted if neither the MFS reference library nor the MFS control block library is to be compressed.

UTPRINT
Is used for messages during the compression of the MFR reference library, and is used for MFS error and status messages during MFS Language utility Phase 2 processing.

The following ddnames refer to data sets used in the MFSRVC procedure. The data set names can be altered to fit installation needs.

REFIN
Refers to the MFS reference library.

FORMAT
Refers to the MFS control block library.

SYSIN
Refers to the input data set, which can be a sequential data set or a member of a partitioned data set.

SYSSNAP
Refers to a data set that is used to receive the output from a SNAP macro if certain severe errors are detected.

SYSPRINT
Refers to the destination of the output. If output is to be sent to a data set (instead of SYSOUT=), use DISP=MOD for the data set.

Related concepts

MFS Device Characteristics table (Application Programming APIs)

Utility control statements and syntax rules

The control statements used by the MFS Language utility are divided into two major categories: definition statements and compilation statements.

The control statements used by the MFS Language utility are divided into two major categories:

- *Definition statements* are used to define message formats, device formats, partition sets, and operator control tables.
- *Compilation statements* are those used to control the compilation and SYSPRINT listings of the definition statements.

Use the definition and compilation control statements to identify a particular function performed by the utility and to specify various options.

The definition and compilation control functions are:

- SYSPRINT LISTING CONTROL

  The following parameters are provided to format the compilation listing: XREF, SUBS, COMP, DIAG, and LINECNT.
- SYSIN and SYSLIB RECORD STACKING and UNSTACKING

  Control statements are provided to allow one or more SYSIN or SYSLIB records to be processed and kept in processor storage for reuse later in the compilation. These statements are an alternative to the COPY facility for groups of statements that are repeated.

  MFLD and DFLD statements can be repetitively generated if preceded by a DO statement and followed by an ENDDO statement. Repetitive DFLD generation supports increments to line and column position information.
- ALPHA CHARACTER GENERATION

  The ALPHA statement allows specification of additions to the set of characters as alphabetic.
- COPY

  The COPY statement allows members of partitioned data sets to be copied into the input stream of the utility preprocessor.

The control statements are written in assembler-like language with the following standard format:

Control statement syntax for the MFS Language utility

label operation operand comments

label
Identifies the statement; if it is shown as optional, it can be omitted. When included, the name must begin in the first position of the statement (column 1) and must be followed by one or more blanks. It can contain from one to eight alphanumeric characters (one to six, for the FMT label), the first of which must be alphabetic.

operation
Identifies the type of control statement. It normally begins in column 10 and must be preceded and followed by one or more blanks.

operand
Is made up of one or more parameters, which can be positional or keyword parameters. A positional parameter in MFS control statements always appears in the first position of the operand, normally starting in column 16. The position of a keyword parameter is not important. The parameters within one operand are separated by commas. In the syntactical description of the control statements, parameters preceded by commas are thus identified as keyword parameters. The operand field itself must be preceded and followed by one or more blanks.

comments
Can be written in a utility control statement, but they must be separated from the last parameter of the operand field by one or more blanks. (If the statement does not include an operand, the comment should be separated from the statement by at least one blank.) A comment line begins with an asterisk in column 1.

Continuation is accomplished by entering a nonblank character in column 72. If the current line is a comment, then the continuation line can begin in any column.

Other considerations are as follows:

- There is no limit on the number of continuation lines.
- There is no limit on the number of characters in the operand field. Individual operand items cannot exceed 256 characters, excluding trailing and embedded second quote characters.
- If a nonstandard character is detected in a literal, a severity 4 warning message is issued. The nonstandard character is retained in the literal.
- If the current line is a control statement, the continuation line must begin in column 16.
- A single ampersand is needed to generate one ampersand character in the literal.

In addition to the definition and compiler statement specifications, several parameters can be specified in the EXEC statement PARM keyword to control the current compilation for the preprocessor and phase 1; one parameter can be specified for phase 2.

The five special rules that follow use actual MFS code as examples.

1. If you code a statement such that an equal sign or a left parenthesis immediately precedes a comma, you can omit the comma.

   ,FTAB=(,FORCE) could be coded as ,FTAB=(FORCE)

2. If you code a statement such that an equal sign immediately precedes a single item enclosed in parentheses, you can omit the parentheses.

   ,FTAB=(,FORCE) could be coded as ,FTAB=,FORCE

3. You can apply both Rule 1 and Rule 2, in either order, to a single item.

   ,FTAB=(,FORCE) could be coded as ,FTAB=FORCE

4. Under no condition can you specify a keyword without specifying at least one parameter immediately after that keyword.

   Neither ,FTAB= nor ,FTAB=,LDEL='**' is permitted.

5. Blanks are required between labels and statement type names, and between statement type names and their parameters; they are not permitted elsewhere unless explicitly represented by the symbol ƀ.

   DEV ƀPAGE is correct, but DEV,PAGE and ƀFTAB= (ƀMIX) are incorrect.

Syntax errors

The MFS Language utility attempts to recover from syntax errors in source statements. No guarantee exists for the correctness of the assumptions made in the recovery, and these assumptions can differ in different releases of IMS. Assumptions made during recovery are based on (1) what is expected when the incorrect item is encountered; (2) what could appear to the right of the item preceding the incorrect item; and (3) what could appear to the left of the incorrect item.

During the process of error recovery, the following notation can be used in the diagnostic messages:

;
: Indicates that the end of the source statement was encountered. The position marker points to the position immediately following the last source item scanned.

L
: Refers to a literal operand item.

V
: Refers to an identifier operand item (alphabetic character optionally followed by alphanumeric characters).

I
: Refers to a numeric operand item.

A
: Refers to an alphanumeric operand item (numeric character optionally followed by alphanumeric characters).

D
: Refers to a delimiter operand item.

Most error recovery messages have a severity code of 4, indicating a warning level error. When an item is deleted, or the syntax scan is aborted, the statement cannot be validly processed and a severity code of 8 is generated.

Invalid sequence of statements

The language utility preprocessor routines that process MSG, FMT, PDB, or TABLE definition statements are organized hierarchically. A routine for a given level processes a statement at that level, reads the next statement, then determines which routine will next receive control.

If the statement just read is the next lower level statement (for example, a DIV statement following a DEV statement), the next lower level routine (for example, the DIV statement processor) is called.

If the statement just read is not the next lower level statement, control can be passed to one of the following three routines:

- The next lower level routine to assume the missing statement (for example, the DIV processor if a DEV statement is followed by a DPAGE statement)
- The same level routine if the statement just read is of the same level as the processor (for example, a series of DFLD statements)
- The next higher level routine (the calling routine) if the statement just read is not the same or the next lower level (for example, a DEV statement following a DFLD statement, an invalid statement, or a statement out of sequence)

Thus, if the hierarchic structure of a MSG, FMT, PDB, or TABLE definition is invalid or a statement operator is misspelled, case (3) will result in control being returned to successively higher level routines. At the highest level, only a FMT, MSG, TABLE, PDB, or END statement will be accepted by the preprocessor.

Therefore, all statements before the next FMT, PDB, MSG, TABLE or END statement will be flushed (that is, not processed) and flagged with the appropriate error message.

Summary of control statements

The definition of message formats, device formats, partition sets, and operator control tables is accomplished with separate hierarchic sets of definition statements.

Message Definition Statement Set

Is used to define message formats. It includes the following statements:

MSG
Identifies the beginning of a message definition.

LPAGE
Identifies a related group of segment/field definitions.

PASSWORD
Identifies a field or fields to be used as an IMS password.

SEG
Identifies a message segment.

DO
Requests iterative processing of the subsequent MFLD statements.

MFLD
Defines a message field. Iterative processing of MFLD statements can be invoked by specifying DO and ENDDO statements. To accomplish iterative processing, the DO statement is placed before the MFLD statements and the ENDDO after the MFLD statements.

ENDDO
Terminates iterative processing of the preceding MFLD statements.

MSGEND
Identifies the end of a message definition.

Format Definition Statement Set

Is used to define device formats. It consists of the following statements:

FMT
Identifies the beginning of a format definition.

DEV
Identifies the device type and operational options.

DIV
Identifies the format as input, output, or both.

DPAGE
Identifies a group of device fields corresponding to an LPAGE group of message fields.

PPAGE
Identifies a group of logically related records that can be sent to a remote application program at one time.

DO
Requests iterative processing of the subsequent RCD or DFLD statements.

RCD
Identifies a group of related device fields that are sent to a remote application program as a single record.

DFLD
Defines a device field. Iterative processing of DFLD statements can be invoked by specifying DO and ENDDO statements. To accomplish iterative processing, the DO statement is placed before the DFLD statements and the ENDDO after the DFLD statements.

ENDDO
Terminates iterative processing of the previous RCD or DFLD statements.

FMTEND
Identifies the end of a format definition.

Partition Definition Statement Set
Is used to define partition sets (Partition Descriptor Blocks). It consists of the following statements:

PDB
Identifies the beginning of a partition set definition and allows the specification of several parameters that describe it.

PD
Defines a Partition Descriptor, which contains the parameters necessary to describe a partition.

PDBEND
Identifies the end of a partition set definition.

TABLE Definition Statement Set
is used to define operator control tables. It includes the following statements:

TABLE
Identifies the beginning of a table definition.

IF
Defines a conditional test and resulting action.

TABLEEND
Identifies the end of a table definition.

Compilation Statements
Are used for variable functions. Compilation statements that are supported by the MFS Language utility are listed in alphabetic order:

ALPHA
Defines a set of characters to be considered alphabetic for the purpose of defining field names and literals.

COPY
Copies a member of the partitioned data set represented by the SYSLIB DD statement into the input stream of the preprocessor.

DO
Requests iterative processing of MFLD or DFLD definition statements.

EJECT
Ejects SYSPRINT listing to the next page.

END
Defines the end of data for SYSIN processing.

ENDDO
Terminates iterative processing of MFLD, RCD, or DFLD definition statements.

EQU
Equates a symbol with a number, alphanumeric identifier, or literal.

PRINT
Controls SYSPRINT options.

RESCAN
Controls EQU processing.

SPACE
Skips lines on the SYSPRINT listing.

STACK
Delineates one or more SYSIN or SYSLIB records that are to be kept in processor storage for reuse.

TITLE
Provides a title for the SYSPRINT listing.

UNSTACK
Retrieves previously stacked SYSIN or SYSLIB records.

Compilation statements are inserted at logical points in the sequence of control statements. For example, TITLE could be first, and EJECT could be placed before each MSG, FMT, or TABLE statement.

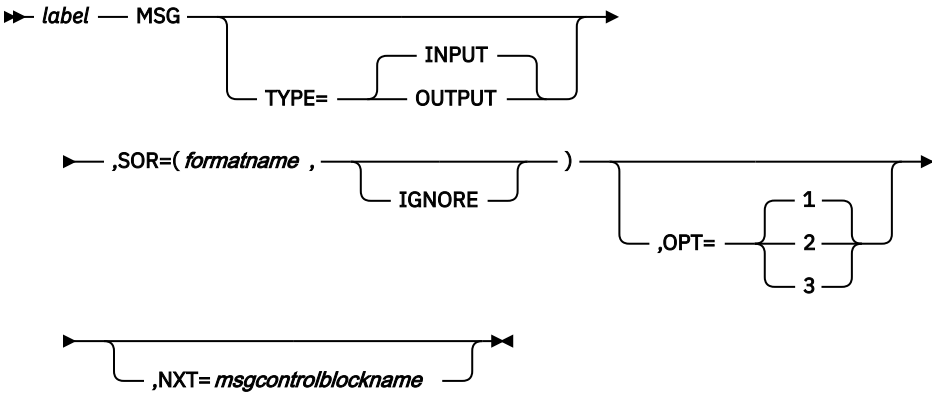
Message definition statements

Message definition statements include the MSG statement, the LPAGE statement, the PASSWORD statement, the SEG statement, the DO statement, the MFLD statement, the ENDDO statement, and the MSGEND statement.

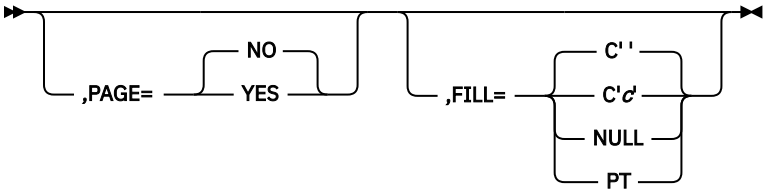
MSG statement

The MSG statement initiates and names a message input or output definition.

Format for MSG TYPE=INPUT or OUTPUT



Format for MSG TYPE=OUTPUT only



Parameters

label
A one- to eight-character alphanumeric name must be specified. This label can be referred to in the NXT operand of another message descriptor.

TYPE=
Defines this definition as a message INPUT or OUTPUT control block. The default is INPUT.

SOR=
Specifies the source name of the FMT statement which, with the DEV statement, defines the terminal or remote program data fields processed by this message descriptor. Specifying IGNORE for TYPE=OUTPUT causes MFS to use data fields specified for the device whose FEAT= operand specifies IGNORE in the device format definition. For TYPE=INPUT, IGNORE should be specified only if the corresponding message output descriptor specified IGNORE. If you use SOR=IGNORE, you must specify IGNORE on both the message input descriptor and the message output descriptor.

OPT=
Specifies the message formatting option used by MFS to edit messages. The default is 1.

NXT=
Specifies the name of a message descriptor to be used to map the next expected message as a result of processing a message using this message descriptor. If TYPE=INPUT, NXT= specifies a message output descriptor. If TYPE=OUTPUT, NXT= specifies a message input descriptor. For ISC output, NXT= becomes the RDPN in the ATTACH FM header.

If TYPE=OUTPUT and the *formatname* specified in the SOR= operand contains formats for 3270 or 3270P device types, the *msgcontrolblockname* referred to by NXT= must use the same *formatname*.

PAGE=
Specifies whether (YES) or not (NO) operator logical paging (forward and backward paging) is to be provided for messages edited using this control block. This operand is valid only if TYPE=OUTPUT. The default is NO, which means that only forward paging of physical pages is provided.

FILL=
Specifies a fill character for output device fields. This operand is valid only if TYPE=OUTPUT. The default is C' '. The fill specification is ignored unless FILL=NONE is specified on the DPAGE statement in the FMT definition. For 3270 output when EGCS fields are present, only FILL=PT or FILL=NULL should be specified. A FILL=PT erases an output field (either a 1- or 2-byte field) only when data is sent to the field, and thus does not erase the DFLD if the application program message omits the MFLD. For DPM-Bn, if OFTAB is specified, FILL= is ignored and FILL=NULL is assumed.

C' '
Character ' ' is the default used to fill device fields. The blank character is interpreted as is X'40' which is a valid printable character.

C'*c*'
Character '*c*' is used to fill device fields. For 3270 display devices, any specification with a value less than X'3F' is changed to X'00' for control characters or to X'40' for other nongraphic characters. For all other devices, any FILL=C'*c*' specification with a value less than X'3F' is ignored and defaulted to X'3F' (which is equivalent to a specification of FILL=NULL).

If you specify C'*c*' as X'36', it changes to either X'0' or X'40' as X'36' is not a valid printable character.

NULL
Specifies that fields are not to be filled. For devices other than 3270 and SLU 2 display, 'compacted lines' are produced when message data does not fill device fields.

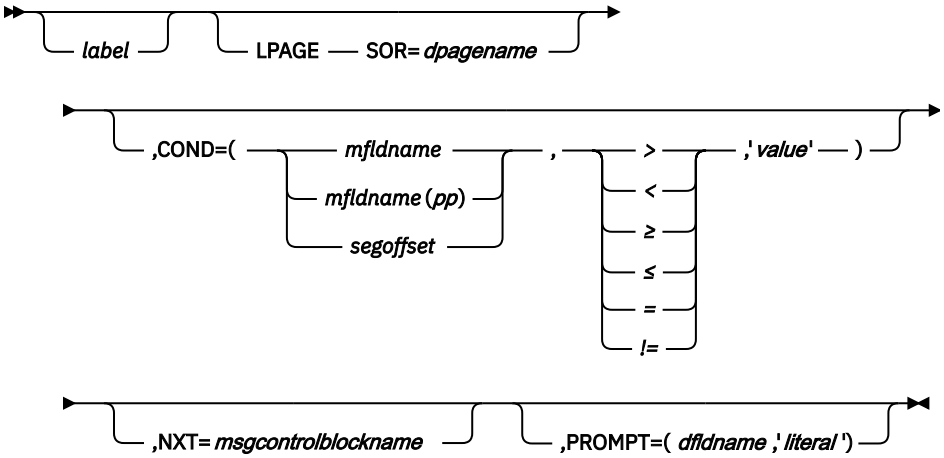
PT
Is identical to NULL except for 3270 and SLU 2 display. For 3270 and SLU 2 display, PT specifies that output fields that do not fill the device field (DFLD) are followed by a program tab character to erase data previously in the field.

Related reference
MFS output message formats (Application Programming)

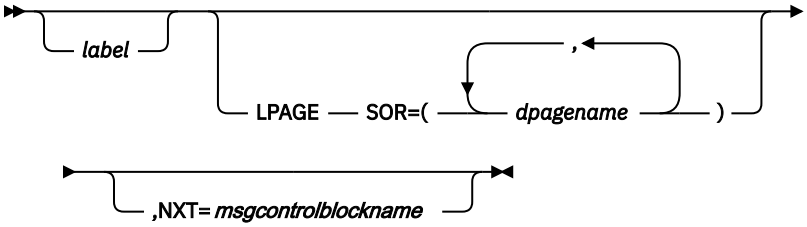
LPAGE statement

The optional LPAGE statement defines a group of segments comprising a logical page.

Format for MSG TYPE=OUTPUT



Format for MSG TYPE=INPUT



Parameters

label
A one- to eight-character alphanumeric name can be specified to uniquely identify this statement.

SOR=
Specifies the name of the DPAGE statement that defines the device format for this logical page. If TYPE=INPUT and more than one DPAGE can be used as a source of data to create an input message, more than one *dpagename* can be specified.

COND=
Describes a conditional test that, if successful, specifies that the segment and field definitions following this LPAGE are to be used for output editing of this logical page. The specified portion of the first segment of a logical page is examined to determine if it is greater than (>), less than (<), greater than or equal to (≥), less than or equal to (≤), equal to (=), or not equal to (≠) the specified literal value to determine if this LPAGE is to be used for editing. COND= is not required for the last LPAGE statement in the MSG definition.

The area examined can be defined by a field name (*mfldname*), an offset in a field (*mfldname(pp)* where *pp* is the offset in the named field), or an offset in the segment (*segoffset*). If the *mfldname(pp)* form is used, *pp* must be greater than or equal to 1. The length of the compare is the length of the specified literal. If OPT=3 is specified on the previous MSG statement, the area to be examined must be within one field as defined on an MFLD statement.

If *segoffset* is used, it is relative to zero, and the specification of that offset must allow for LLZZ of the segment (that is, the first data byte is at offset 4).

If *pp* is used, the offset is relative to 1 with respect to the named field (that is, the first byte of data in the field is at offset 1, not zero).

If the *mfldname* specified is defined with ATTR=YES, the *pp* offset must be used. The minimum offset specified must be 3. That is, the first byte of data in the field is at offset 3, following the two bytes of attributes.

If ATTR=*nn* is specified, the minimum offset must be one plus twice *nn*. Thus, if ATTR=2 is specified, *pp* must be at least 5, and, if ATTR=(YES,2) is specified, *pp* must be at least 7.

If the conditional tests for all LPAGEs fail, the last LPAGE in this MSG definition is used for editing.

If LPAGE selection is to be specified using the command data field, that is, **/FORMAT***modname... (data)*, the MFLD specified in the LPAGE COND=*mfldname* parameter should be within the first 8 bytes of the associated LPAGEs of the MOD.

NXT=
Specifies the name of the message descriptor to be used to map the next message if this logical page is processed. This name overrides any NXT=*msgcontrolblockname* specified on the preceding MSG statement.

PROMPT=
Specifies the name of the DFLD into which MFS should insert the specified literal when formatting the last logical page of an output message. If FILL=NULL is specified once the prompt literal is displayed, it can remain on the screen if your response does not cause the screen to be reformatted.

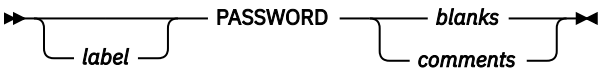
PASSWORD statement

The PASSWORD statement identifies one or more fields to be used as an IMS password or password phrase.

When used, the PASSWORD statement and its associated MFLDs must precede the first SEG statement in an input LPAGE or MSG definition. Up to 8 MFLD statements can be specified after the PASSWORD statement but the total password length must not exceed 100 characters. The fill character must be X'40'. For option 1 and 2 messages, the first 8 characters of data after editing are used for the IMS password. For option 3 messages, the data content of the first field after editing is used for the IMS password.

A password for 3270 input can also be defined in a DFLD statement. If both password methods are used, the password specified in the MSG definition is used.

Format



Parameters

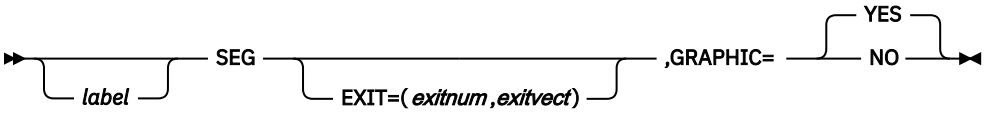
label
A one- to eight-character alphanumeric name can be specified to uniquely identify this statement.

SEG statement

The SEG statement delineates message segments and is required only if multisegment message processing is required by the application program.

Output message segments cannot exceed your specified queue buffer length. Only one segment should be defined for TYPE=INPUT MSGs when the input message destination is defined as a single segment command or transaction. If more than one segment is defined, and the definition is used to input a single segment command or transaction, care must be used to ensure that your input produces only one segment after editing.

Format for MSG TYPE=INPUT



Format for MSG TYPE=OUTPUT

Parameters

label
A 1- to 8-character name can be specified to uniquely identify this statement.

EXIT=
Describes the segment edit exit routine interface for this message segment. *exitnum* is the exit routine number and *exitvect* is a value to be passed to the exit routine when it is invoked for this segment. *exitnum* can range from 0 to 127. *exitvect* can range from 0 to 255. Unless NOSEGEXIT is specified on the DIV statement (for DPM devices only), the SEG exit is invoked when processing completes for the input segment.

GRAPHIC=
Specifies for MSG TYPE=INPUT whether (YES) or not (NO) IMS should perform upper case translation on this segment if the destination definition requests it (see the EDIT= parameter of the TRANSACT or NAME macro). The default is YES. If input segment data is in nongraphic format (packed decimal, EGCS, binary, and so forth), GRAPHIC=NO should be specified. When GRAPHIC=NO is specified, FILL=NULL is invalid for MFLDs within this segment.

The following list shows the translation that occurs when GRAPHIC=YES is specified and the input message destination is defined as requesting upper case translation:

Before Translation
After Translation

a through z
A through Z

X'81' through X'89'
X'C1' through X'C9'

X'91' through X'99'
X'D1' through X'D9'

X'A2' through X'A9'
X'E2' through X'E9'

If FILL=NULL is specified for any MFLD in a segment defined as GRAPHIC=YES, the hexadecimal character X'3F' is compressed out of the segment. If GRAPHIC=NO and FILL=NULL are specified in the SEG statement, any X'3F' in the non-graphic data stream is compressed out of the segment and undesirable results might be produced. Non-graphic data should be sent on output as fixed length output fields and the use of FILL=NULL is not recommended in this case.

For MSG TYPE=OUTPUT, the GRAPHIC= keyword applies only for DPM. It specifies whether (YES) or not (NO) nongraphic control characters (X'00' to X'3F') in the data from the IMS application program are to be replaced by blanks. The default value is YES. If NO is specified, MFS allows any bit string received from an IMS application program to flow unmodified through MFS to the remote program.

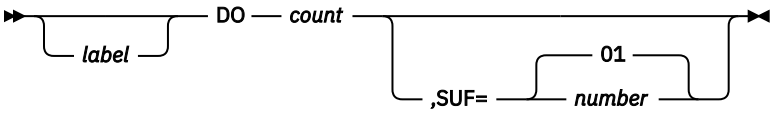
Restriction: When GRAPHIC=NO is specified, IMS application programs using Options 1 and 2 cannot omit segments in the middle of an LPAGE, or truncate or omit fields in the segment using the null character (X'3F').

DO statement

The DO statement causes repetitive generation of MFLD statements between the DO and ENDDO statements.

DO is optional, but a message that includes a DO must include a subsequent ENDDO.

Format



Parameters

label
A one- to eight-character alphanumeric name can be specified. It is not used.

count
Specifies how many times to generate the following MFLD statements. The maximum *count* that can be specified is 99; if more than 99 is specified, the 2 rightmost digits of the specified *count* are used (for example, 03 would be used if 103 were specified) and an error message is issued.

SUF=
Specifies the 1- or 2-digit suffix to be appended to the MFLD *label* and *dfldname* of the first group of generated MFLD statements. The default is 01. MFS increases the suffix by 1 on each subsequent generation of statements.

If the specified suffix exceeds 2 digits, MFS uses the rightmost 2 digits.

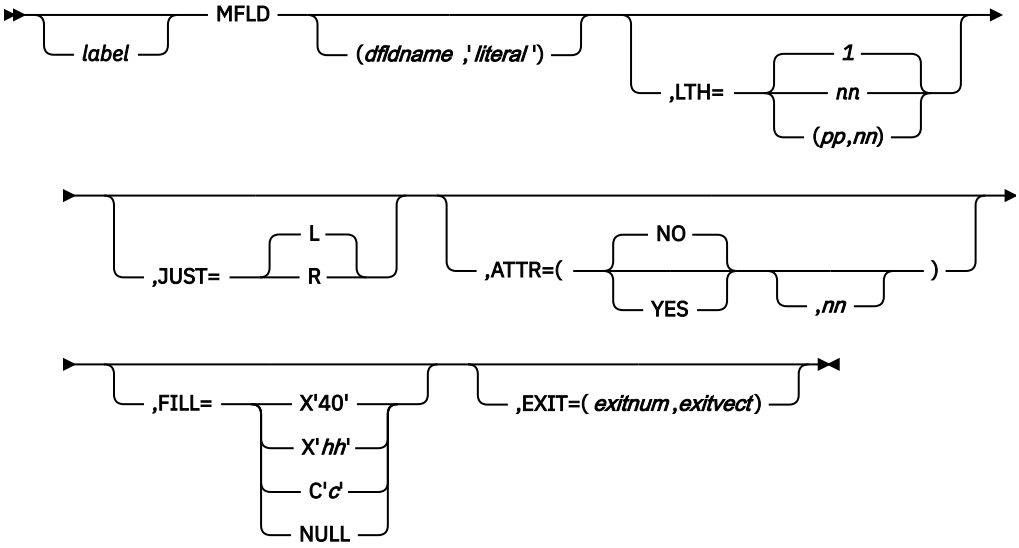
If the specified *count* is such that the generated suffix eventually exceeds 2 digits, MFS reduces the *count* to the largest legitimate value. For example, if *count* equals 8 and SUF=95, invalid suffixes of 100, 101, and 102 would result. In this instance, MFS reduces *count* to 5, processes the statement, and issues an error message.

MFLD statement

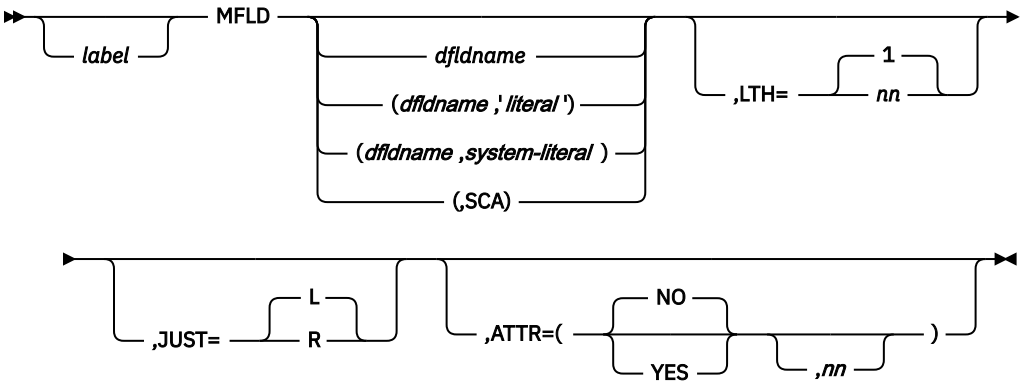
The MFLD statement defines a message field as it will be presented to an application program as part of a message output segment.

At least one MFLD statement must be specified for each MSG definition.

Format for MSG TYPE=INPUT



Format for MSG TYPE=OUTPUT



Parameters

label

A one-to eight-character alphanumeric name can be specified. *label* is required if it is referred to in the COND operand of the previous LPAGE statement. It can be used to uniquely identify this statement. If the MFLD is between the DO and ENDDO statements, *label* is restricted to 6 characters or less. DO statement processing appends a 2-digit suffix (a sequence number, 01 to 99) to the label and prints the label as part of the generated MFLD statement. If *label* is more than 6 characters and iterative generation is used, the label is truncated at 6 characters, and the 2-digit sequence number is added to make the 8-character name. No error message is issued if this occurs.

dfldname

Specifies the device field name (defined using the DEV or DFLD statement) from which input data is extracted or into which output data is placed. If this parameter is omitted when defining a message output control block, the data supplied by the application program is not displayed on the output device. If the repetitive generation function of MFS is used (DO and ENDDO statements), *dfldname* should be restricted to 6 characters maximum length. When each repetition of the statement is generated, a 2-character sequence number (01 to 99) is appended to *dfldname*. If the *dfldname* specified here is greater than 6 bytes and repetitive generation is used, *dfldname* is truncated at 6 characters and a 2-character sequence number is appended to form an 8-character name. No error message is provided if this occurs. This parameter can be specified in one of the following formats:

dfldname
Identifies the device field name from which input data is extracted or into which output data is placed.

'literal'
Can be specified if a literal value is to be inserted in an input message.

(*dfldname,'literal'*)
If TYPE=OUTPUT, this describes the literal data to be placed in the named DFLD. When this form is specified, space for the literal must not be allocated in the output message segment supplied by the application program.

If TYPE=INPUT, this describes the literal data to be placed in the message field when no data for this field is received from the device. If this *dfldname* is used in the PFK parameter of a DEV statement, this literal is always replaced by the PF key literal or control function. However, when this *dfldname* is specified in the PFK parameter, but the PF key is not used, the literal specified in the MFLD statement is moved into the message field. When physical paging is used, the literal is inserted in the field but is not processed until after the last physical page of the logical page has been displayed.

In both cases, if the LTH= operand is specified, the length of the literal is truncated or padded as necessary to the length of the LTH= specification. If the length of the specified literal is less than the defined field length, the literal is padded with blanks if TYPE=OUTPUT and with the specified fill character (FILL=) if TYPE=INPUT. If no fill character is specified for input, the literal is padded with blanks (the default). The length of the literal value cannot exceed 256 bytes.

(*dfldname,system-literal*)
Specifies a name from a list of system literals. A system literal functions like a normal literal except that the literal value is created during formatting prior to transmission to the device. The LTH=, ATTR=, and JUST= operands cannot be specified. When this form is specified, space for the literal must not be allocated in the output message segment supplied by the application program.

The following table shows the system literals and their associated lengths and formats.

Table 16. Lengths and formats of system literals

	Produces literal of:		
System literal name	**Length**	**Format**	**Notes**
LTSEQ	5	*nnnnn*	1
LTNAME	8	*aaaaaaaa*	1
TIME	8	HH:MM:SS	
DATE1 or YYDDD	6	YY.DDD	
DATE2 or MMDDYY	8	MM/DD/YY	
DATE3 or DDMMYY	8	DD/MM/YY	
DATE4 or YYMMDD	8	YY/MM/DD	
DATE1Y4 or YYYYDDD or DATEJUL	8	YYYY.DDD	
DATE2Y4 or MMDDYYYY or DATEUSA	10	MM/DD/YYYY	
DATE3Y4 or DDMMYYYY or DATEEUR	10	DD/MM/YYYY	

Table 16. Lengths and formats of system literals (continued)

	Produces literal of:		
System literal name	**Length**	**Format**	**Notes**
DATE4Y4 or YYYYMMDD or DATEISO	10	YYYY/MM/DD	
LPAGENO	4	*nnnn*	2
LTMSG	14	MSG WAITING Q*x*	3

Notes:

1. LTSEQ is the output message sequence number for the logical terminal. The value created is the logical terminal dequeue count plus 1. The first output message after an IMS cold start or **/NRESTART BUILDQ** has a sequence number of 00001. Certain IMS-created messages do not change this number.

   LTNAME is the logical terminal (LTERM) name of the LTERM for which this message is being formatted.

   Messages generated by the IMS control region in response to terminal input (error messages, most command responses) do not have an LTSEQ or an LTNAME. These messages use the IMS message output descriptor DFSMO1. In these instances, the values provided are 00000 and blanks, respectively.
2. LPAGENO specifies that the current logical page number of the message be provided as a system literal. This number corresponds to the page number you entered for an operator logical page request. The literal produced is a 4-digit number with leading zeros converted to blanks.
3. LTMSG specifies that when this output message is sent to the terminal, the literal 'MSG Waiting Q*x*' (where *x* is message queue number 1, 2, 3, or 4) is sent in the LTMSG field if there are messages in the queue for the terminal. If there are no messages in the queues, other than the current queue, blanks are sent in the LTMSG field.

   Usually the message waiting is sent when the current message is dequeued. If the message is waiting in Q1, it is sent. If the message is in Q2 and the terminal is in exclusive mode, it is sent (when any other messages from Q1 are sent). If the message is in Q2 and conversational status does not prevent it from being sent or if the message is in Q3 or Q4 and the exclusive or conversational status does not prevent it from being sent, it is sent. If a message is waiting to be sent on another queue and the terminal is in conversation, the conversation can be held to view the message; if the terminal is in exclusive mode, the message can be viewed when the terminal is taken out of exclusive mode. If you are entering response mode transactions, the message can be viewed before entering response mode transaction input from the terminal.

   This system literal is recommended for conversational mode. It is not recommended for ISC subsystems.

(,SCA)

Defines this output field as the system control area which is not displayed on the output device. There can be only one such field in a logical page (LPAGE) and it must be in the first message segment of that page. If no logical pages are defined, only one SCA field can be defined and it must be in the first segment of the output message. This specification is valid only if TYPE=OUTPUT was specified on the previous MSG statement.

LTH=

Specifies the length of the field to be presented to an application program on input or received from an application program on output. Default or minimum value is 1. Maximum value is 8000. (The maximum message length must not exceed 32767.)

The form (*pp*,*nn*) can be used when defining an input field; however, a field name must be specified in the first positional parameter if the (*pp*,*nn*) form is used. The value supplied for *pp* specifies which byte in the input data field is to be considered the first byte of data for the message field. For example, a *pp* of 2 specifies that the first byte of input data is to be ignored, and the second byte becomes the first byte of this field. The value of *pp* must be greater than or equal to 1. The value supplied for *nn* specifies the length of the field to be presented to an application program.

If (,SCA) is specified in the positional parameter, the specified LTH= value must be at least 2.

LTH= can be omitted if a literal is specified in the positional operand (TYPE=INPUT), in which case, length specified for *literal* is used. If LTH= is specified for a literal field, the specified literal is either truncated or padded with blanks to the specified length. If the MFLD statement appears between a DO and an ENDDO statement, a length value is printed on the generated MFLD statement, regardless of whether LTH= is specified in the MFLD source statement.

JUST=

Specifies that the data field is to be left-justified (L) or right-justified (R) and right- or left- truncated as required, depending upon the amount of data expected or presented by the device format control block. The default is L.

ATTR=

Specifies whether (YES) or not (NO) the application program can modify the 3270 attributes and the extended attributes (*nn*).

If YES, 2 bytes must be reserved for the 3270 attribute data to be filled in by the application program on output and to be initialized to blanks on input. These 2 bytes must be included in the LTH= specification.

The value supplied for *nn* is the number of extended attributes that can be dynamically modified. The value of *nn* can be a number from 1 to 6. An invalid specification will default to 1. Two additional bytes per attribute must be reserved for the extended attribute data to be filled in by the application program on output and to be initialized to blanks on input. These attribute bytes must be included in the MFLD LTH= specification.

The following example shows valid specifications for ATTR= and the number of bytes that must be reserved for each different specification:

MFLD ,ATTR=(YES,*nn*)
2 + (2 × *nn*)

MFLD ,ATTR=(NO,*nn*)
2 × *nn*

MFLD ,ATTR=(*nn*)
2 × *nn*

MFLD ,ATTR=YES
2

MFLD ,ATTR=NO
0

ATTR=YES and *nn* are invalid if a literal value has been specified through the positional parameter in an output message.

The attributes in a field sent to another IMS ISC subsystem are treated as input data by MFS regardless of any ATTR= specifications in the format of the receiving subsystem. For example, a message field (MFLD) defined as ATTR=(YES,1),LTH=5 would contain the following:

```
00A0C2F1C8C5D3D3D6
```

If the MFLD in the receiving subsystem is defined as LTH=9 and without ATTR=, the application program receives:

```
00A0C2F1C8C5D3D3D6
```

If the MFLD in the receiving subsystem is defined as LTH=13 and ATTR=(YES,1), the application program receives:

```
4040404000A0C2F1C8C5D3D3D6
```

If the MFLD in the receiving subsystem is defined as LTH=5 and ATTR=(YES,1), the application program receives:

```
4040404000A0C2F1C8
```

The input SEG statement should be specified as GRAPHIC=NO to prevent translation of the attribute data to uppercase.

FILL=
Specifies a character to be used to pad this field when the length of the data received from the device is less than the length of this field. This character is also used to pad when no data is received for this field (except when MSG statement specifies option 3.) This operand is only valid if TYPE=INPUT. The default is X'40'.

X'*hh*'
Character whose hexadecimal representation is *hh* is used to fill fields. FILL=X'3F' is the same as FILL=NULL.

C'*c*'
Character *c* is used to fill fields.

NULL
Causes compression of the message segment to the left by the amount of missing data in the field.

EXIT=
Describes the field edit exit routine interface for this message field. The exit routine number is specified in *exitnum*, and *exitvect* is a value to be passed to the exit routine when it is invoked for this field. The value of *exitnum* can range from 0 to 127. The value of *exitvect* can range from 0 to 255. The address of the field as it exists after MFS editing, (but before NULL compression for option 1 and 2), is passed to the edit exit routine, along with the vector defined for the field. (If NOFLDEXIT is specified for a DPM device, the exit routine will not be invoked.) The exit routine can return a code with a value from 0 to 255. MFS maintains the highest such code returned for each segment for use by the segment edit routine. EXIT= is invalid if '*literal*' is specified on the same MFLD statement.

Printing generated MFLD statements

The generated MFLD statements can be printed in a symbolic source format by specifying COMP in the parameter list of the EXEC statement.

This provides a means of seeing the results of the MFLD statement generation without having to interpret the intermediate text blocks.

The following items are printed for each generated MFLD statement:

- The generated statement sequence number followed by a + (plus sign) to indicate that the MFLD statement was generated as a result of DO statement processing.
- The MFLD statement label, if present, including the appended suffix.
- The statement operator, MFLD.
- *dfldname*, if present, including the appended suffix.
- For ECGS literals, the G, SO, and SI is not present. Literals are truncated if there is insufficient room to print all specifications. Truncation is indicated by a portion of the literal followed by an ellipsis (...) representing the truncated portion.

- The system literal name, if present.

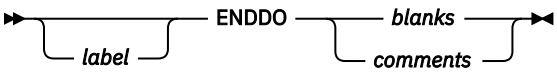
  If both *dfldname* and a literal are present, they are enclosed in parentheses.
- (,SCA), if present.
- The field length, in the form LTH=*nnnn* (or LTH=(*pppp,nnnn*), if present).
- JUST=L or R, if present.
- ATTR=YES, if present.
- ATTR=*nn*, if present.

No other operands are printed, even if specified on the source MFLD statement.

ENDDO statement

The ENDDO statement terminates the group of MFLD statements that are to be repetitively generated.

The generated MFLD statements are printed immediately following the ENDDO statement. ENDDO is required when a DO statement has been specified.



label
A one- to eight-character alphanumeric name can be specified. It is not used.

MSGEND statement

The MSGEND statement terminates a message input or output definition and is required as the last statement in the definition.

If this is the end of the job submitted, it must also be followed by an END compilation statement.

label
A one- to eight-character alphanumeric name can be specified. It is not used.

Format definition statements

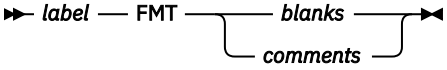
Format definition statements include the FMT statement, the DEV statement, the DIV statement, the DPAGE statement, the PPAGE statement, the DO statement, the RCD statement, the DFLD statement, the ENDDO statement, and the FMTEND statement.

FMT statement

The FMT statement initiates and names a format definition that includes one or more device formats differing only in the device type and features specified in the DEV statement.

Each device format included in the format definition specifies the layout for data sent to or received from a device or a remote program.

Format



Parameters

label

A required one- to six-character alphanumeric name that is referred to by message descriptors in the SOR= operand of MSG statements.

The name specified for *label* becomes part of the member name used for the resulting device output format and device input format blocks that are stored in the IMS.FORMAT library.

If DEV TYPE=DPM-An, and DIV OPTIONS=MSG, the name specified for *label* is sent to the remote program as the data name in the output message header.

If DEV TYPE=DPM-Bn, and DIV OPTIONS=(MSG,DNM), the name specified for *label* is sent to the other subsystem as the data structure name in the DD FM header.

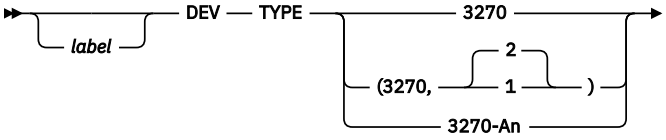
DEV statement

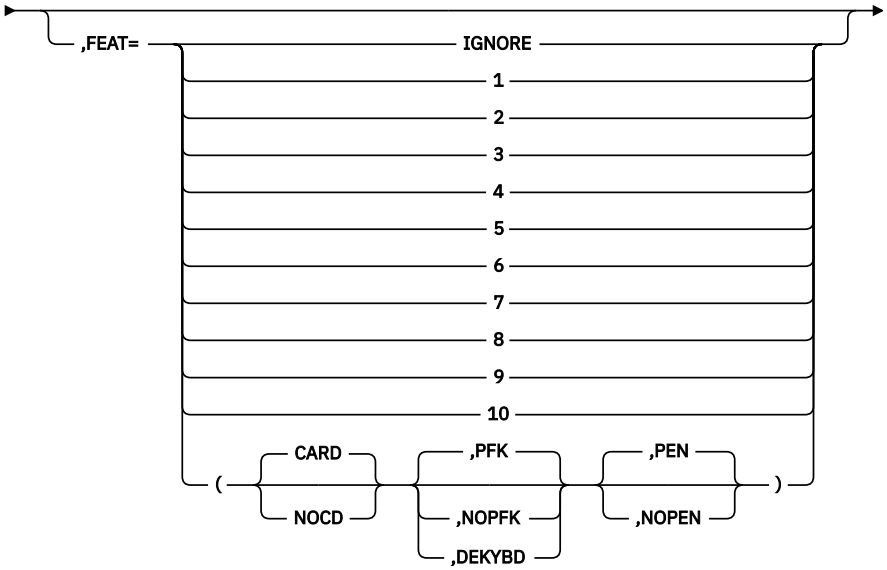
The DEV statement defines device characteristics for a specific device or data formats for a specific device type.

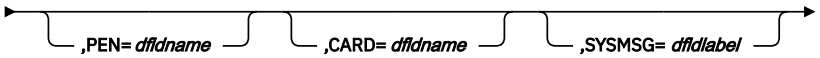
The DFLD statements following this DEV statement are mapped using the characteristics specified until the next DEV or FMTEND statement is encountered. For DPM devices, the DEV statement specifies the DPM program type number and (optionally) a feature set number.

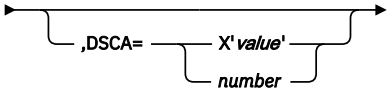
Important: Read the TYPE= operand description before using the DEV statement.

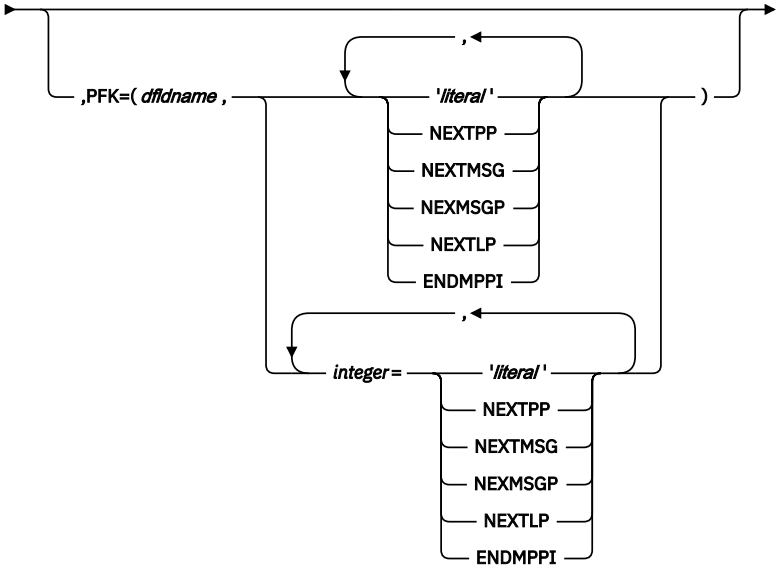
Format for 3270 display

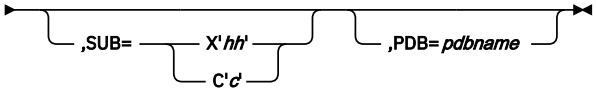




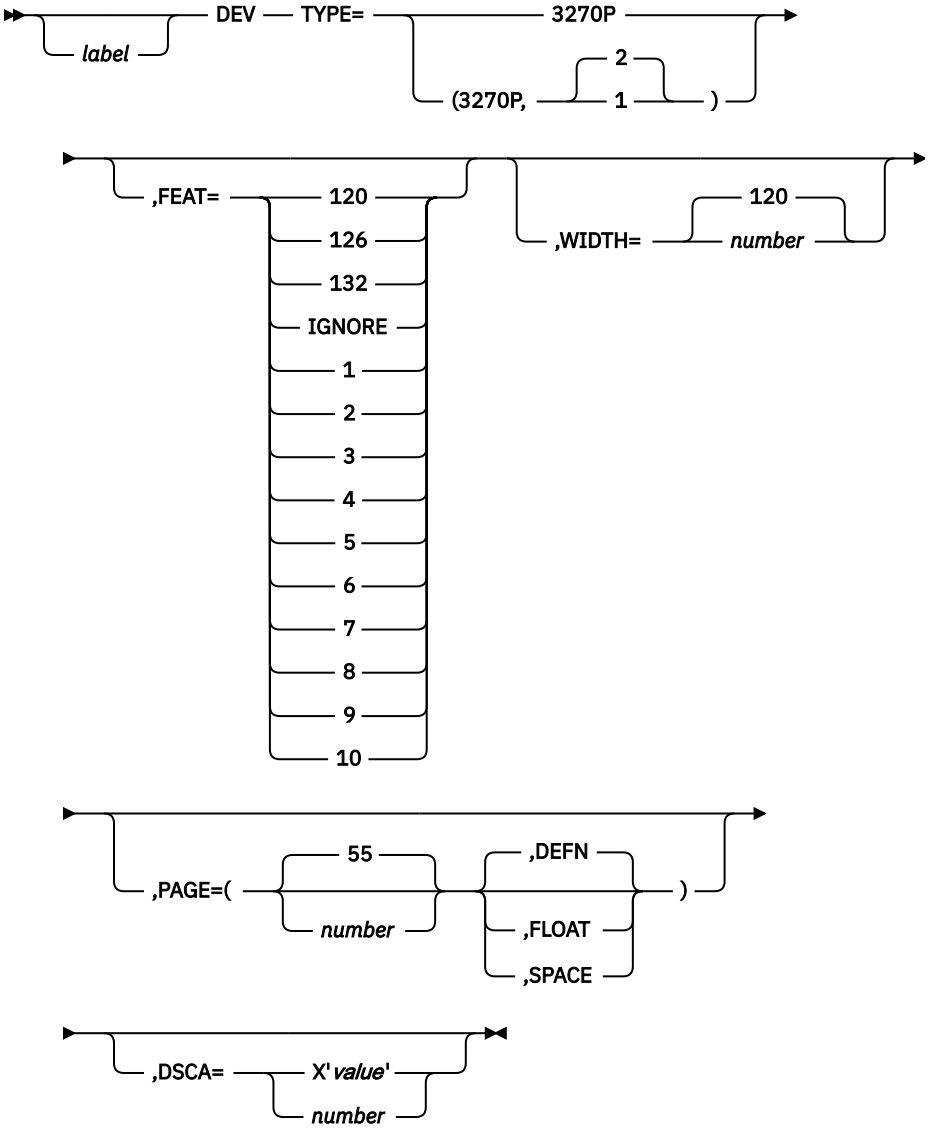




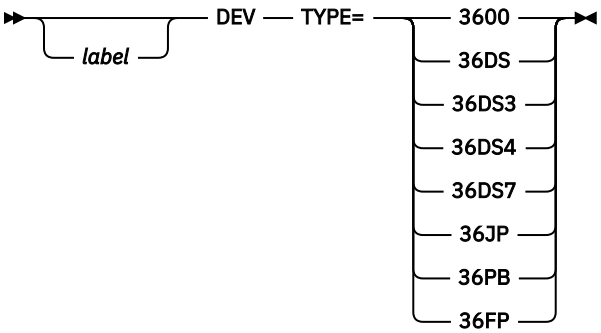




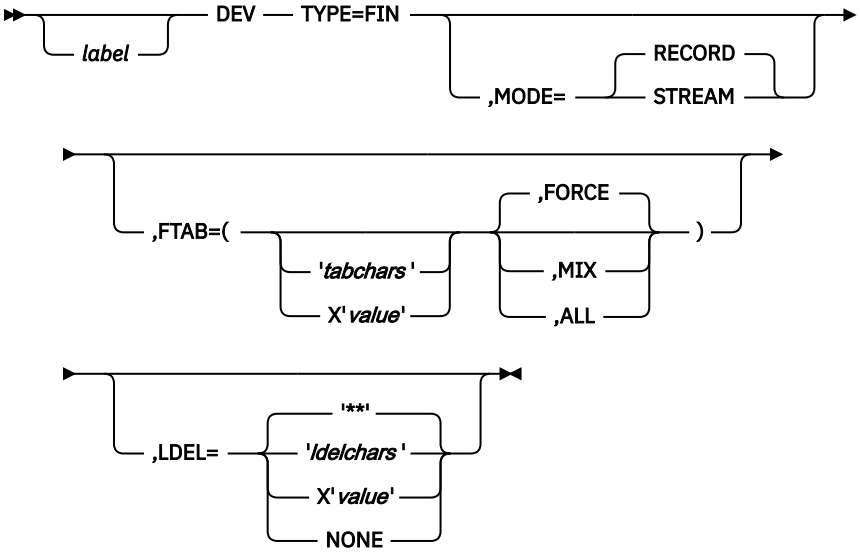
Format for 3270 printers



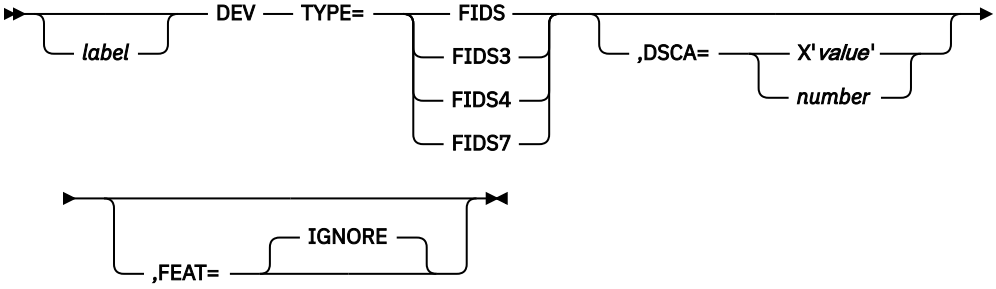
Format for Finance workstations (3600 or 4700)



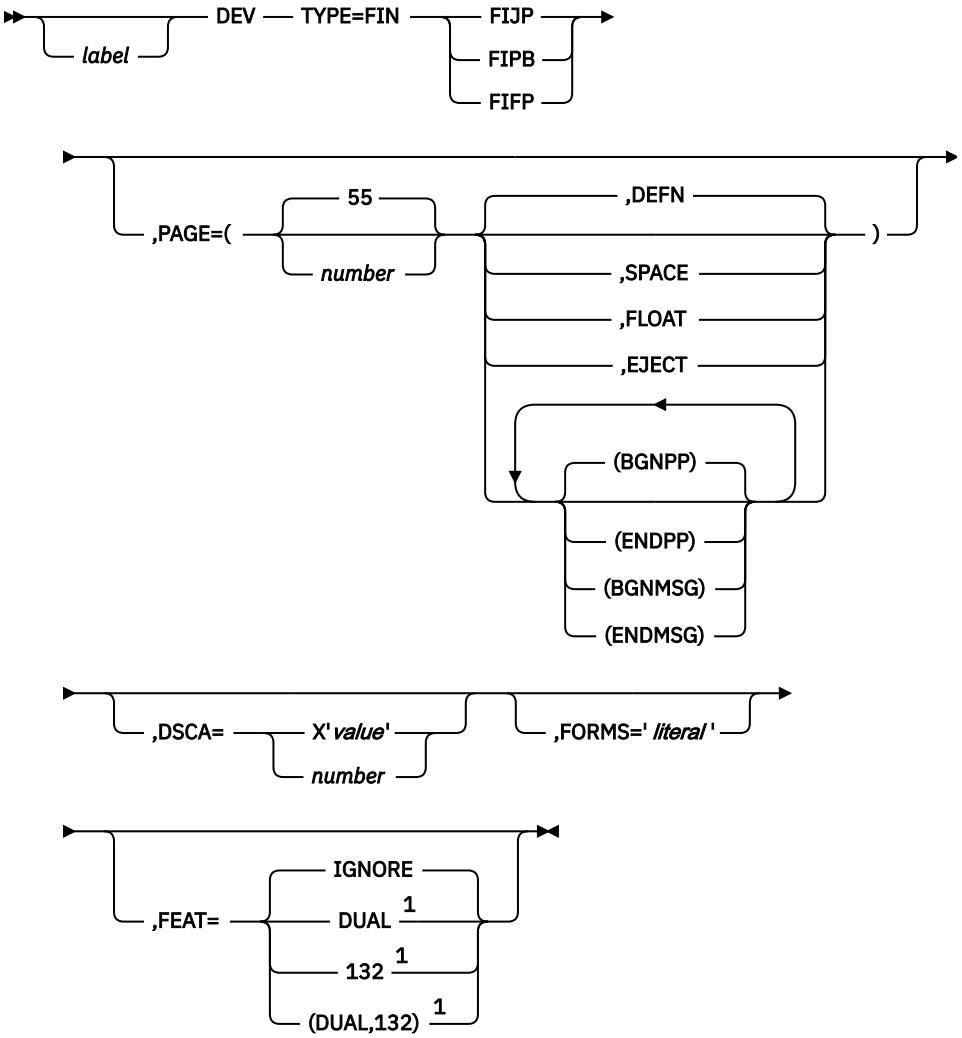
Format for DEV TYPE=FIN



Format for DEV TYPE=FIDS, FIDS3, FIDS4, FIDS7



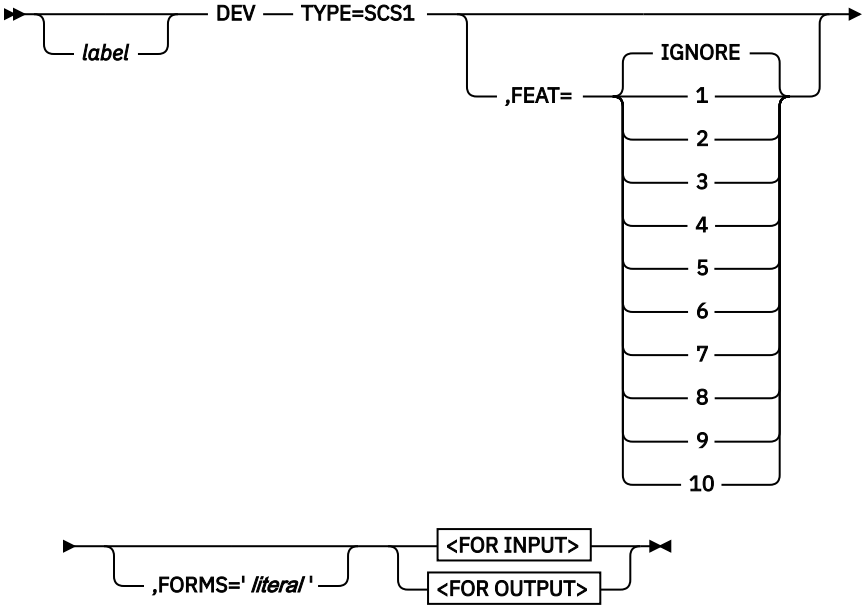
Format for DEV TYPE=FIJP, FIPB, FIFP



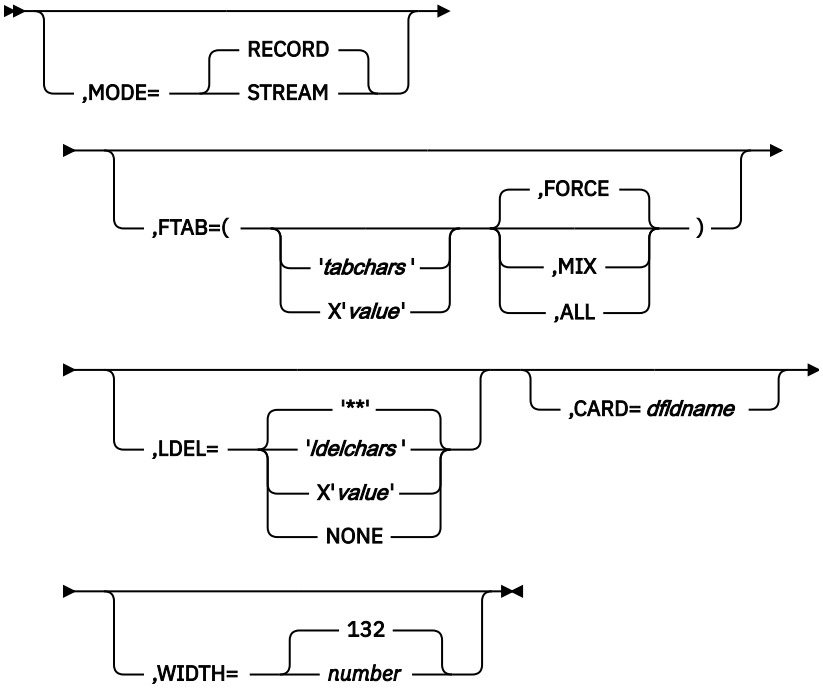
Notes:

[1] FIFP only

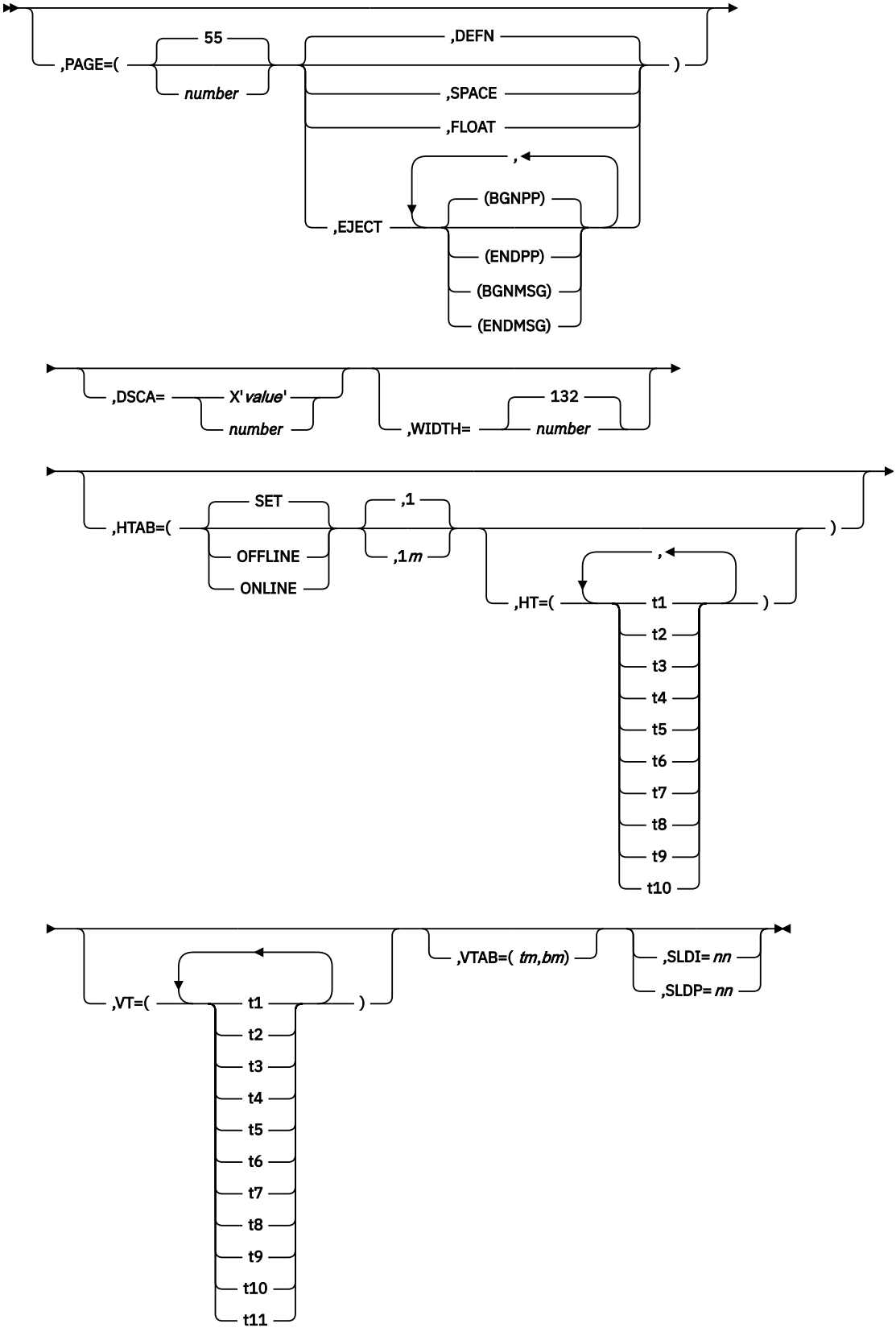
Format for SCS1



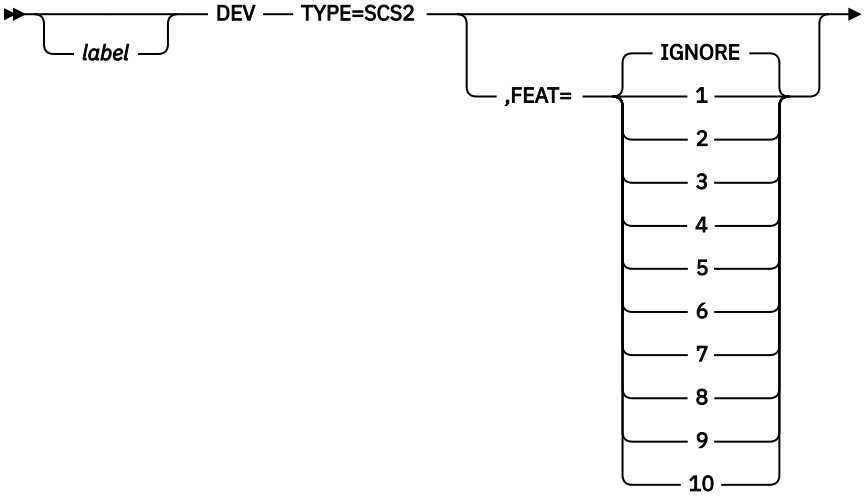
<FOR INPUT>

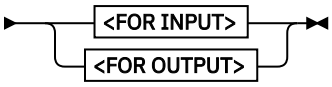


<FOR OUTPUT>

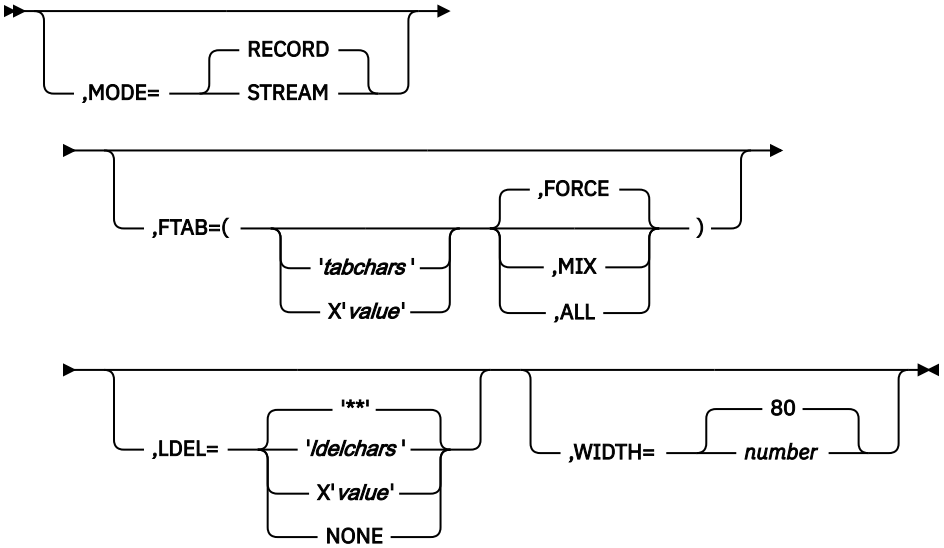

,PAGE=(
55
number
,DEFN
,SPACE
,FLOAT
,EJECT
(BGNPP)
(ENDPP)
(BGNMSG)
(ENDMSG)
)
,DSCA=
X'value'
number
,WIDTH=
132
number
,HTAB=(
SET
OFFLINE
ONLINE
,1
,1m
,HT=(
t1
t2
t3
t4
t5
t6
t7
t8
t9
t10
)
,VT=(
t1
t2
t3
t4
t5
t6
t7
t8
t9
t10
t11
)
,VTAB=(tm,bm)
,SLDI= nn
,SLDP= nn

Format for SCS2

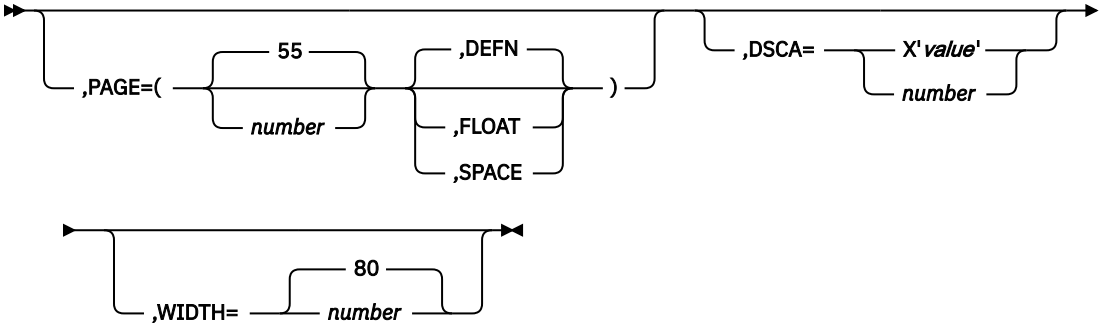




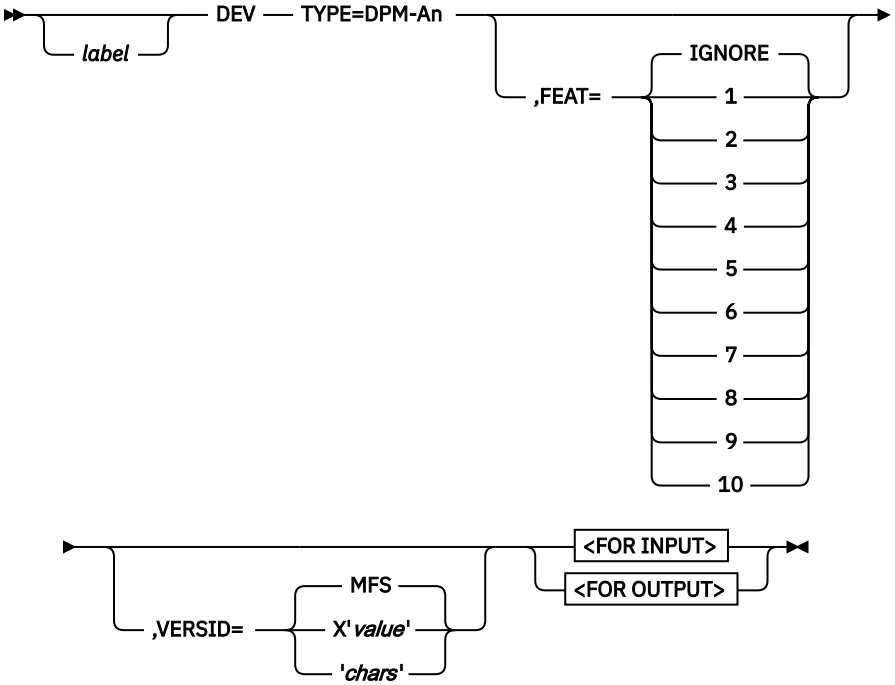
<FOR INPUT>



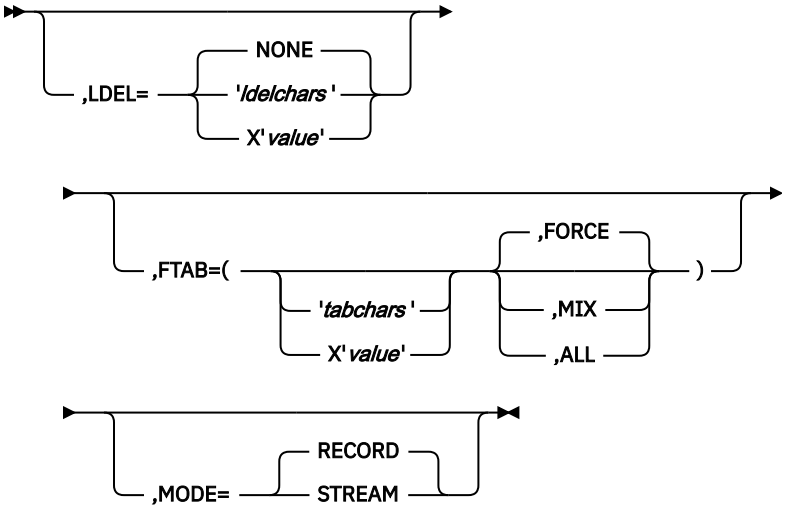
<FOR OUTPUT>



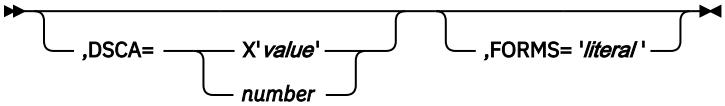
Format for DPM-An



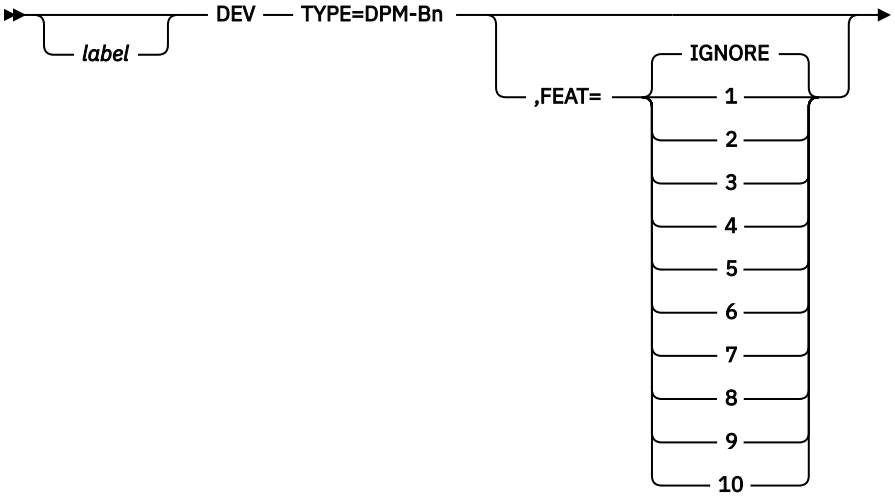
<FOR INPUT>

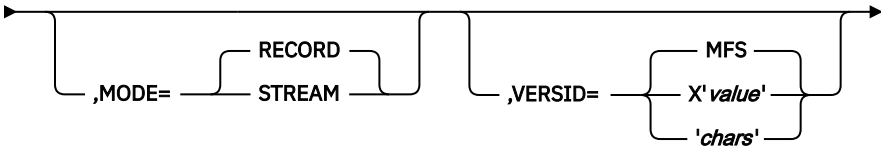


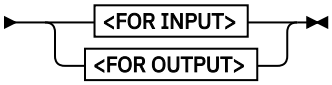
<FOR OUTPUT>



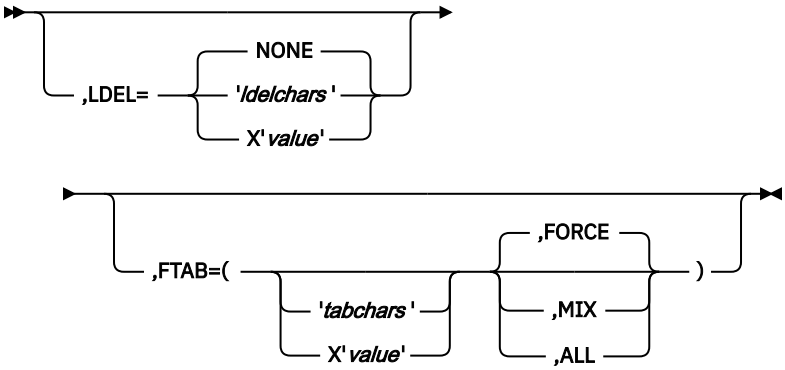
Format for DPM-Bn



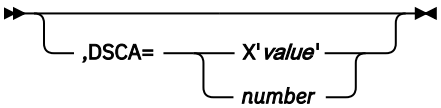




<FOR INPUT>



<FOR OUTPUT>



Parameters

label
An optional one- to eight-character alphanumeric name that uniquely identifies this statement.

TYPE=
Specifies the device type and model number of a device using this format description. The 3284-3 printer attached to a 3275 is supported only as TYPE=3270P. The model number specified when defining a format for a 3284-3 is the model number of the associated 3275.

TYPE=3270-An specifies a symbolic name for 3270 and SLU 2 displays with the screen size defined during IMS system definition, feature numbers *n*=1-15. This specification causes the MFS Language utility to read the MFS device characteristics table (DFSUDT0x) to extract the screen size.

TYPE=DPM-Bn specifies the device as an ISC node. The device type specified by *n* must agree with the specification of the component (COMPT=) on the system definition TERMINAL macro.

Based on the device and model used, specify:

TYPE=
Device-Model

3270, 1
3275-1

3276-1,11 (defined at IMS system definition as 3270 model 1)

3277-1

3278-1 (defined at IMS system definition as 3277 model 1)

SLU 2 (480 characters)

3270,2
3275-2 SLU 2 (1920 characters)

(any display defined during IMS system definition as 'mod 2' with screen area of 1920 characters)

3270-An
3270-An (applies to any 3270 or SLU 2 display defined as TYPE=3270-An during IMS system definition)

Examples of 3270 devices that can be defined as 3270-An and the recommended standard of associating screen sizes with the device type symbolic name follow:

Device
Screen size and definition

3180
24×80 screen size defined as 3270-A2

327X-1,11
12×80 screen size defined as 3270-A1

327X-2,12
24×80 screen size defined as 3270-A2

327X-3,13
32×80 screen size defined as 3270-A3

327X-4,14
43×80 screen size defined as 3270-A4

3278-5
27x132 screen size defined as 3270-A7

3290
62x160 screen size defined as 3270-A8

or

24×80 screen size defined as 3270-A2

5550
3270 Kanji Emulation or 3270 PC with 24×80 screen size defined as 3270-A2

3270P,1
3284-1

3286-1

3287 (with 480 character print feature and not attached as SLU 1 or SLU 4)

3289 (with 480 character print feature and not attached as SLU 1 or SLU 4)

3270P,2
3284-2

3286-2

3287 (with 1920 character print feature and not attached as SLU 1 or SLU 4)

3289 (with 1920 character print feature and not attached as SLU 1 or SLU 4)

FIN
Finance application program (input only)

FIDS
Finance display component (6×40; for example, 3604-1 or -2)

FIDS3
Finance display component (12×40; for example, 3604-3)

FIDS4
Finance display component (16×64; for example, 3604-4)

FIDS7
Finance display component (24×80; for example, 3604-7)

FIJP
Finance journal printer

FIPB
Finance passbook printer

FIFP
Finance administrative printer

SCS1
The following console keyboard printers:

NTO
3771
3773
3774
3775
3776
5553
5557
SLU 1 (with a print data set or bulk printer)
SLU 4
3289 and 3287 when attached to IMS as SLU 1

SCS2
3521 card punch

3501 card reader

2502 card reader

SLU 1 (transmit data set)

SLU 4

DPM-An
SLU P (n is value 1-15)

DPM-Bn
ISC (n is value 1-15)

MODE=
Specifies the manner in which field scanning is to occur. Default value is RECORD. MODE= is valid for DPM-An input only, and for DPM-Bn input and output. For DPM-Bn, if the input and output modes are not the same, each DIV statement must be preceded by a DEV statement.

RECORD
Specifies that fields are defined as occurring within specific records (a line from a device, a transmission from a remote program) that is transmitted from the device or program. For DPM-Bn, Record mode must be specified for variable length, variable blocked (VLVB) format records.

STREAM
Specifies that fields are defined as a contiguous stream of fields—record boundaries do not affect the MFS scan. Fields can be split across records and fields can be entered from any record provided they are entered in the defined sequence. For DPM-Bn, Stream mode must be specified for chained request/response units (RUs).

FTAB=
Specifies the field tab (FTAB) characters that you or a remote program can use to terminate an input field when either the length of the data entered is less than the defined field length, or no data for the field exists:

- For FIN, DPM-An, and DPM-Bn, a maximum of eight FTAB characters or 16 hexadecimal digits can be specified, and at least one character (or two hexadecimal digits) should be specified.
- For SCS1, up to four FTAB characters or eight hexadecimal digits can be specified; the characters NL, LF, HT, and VT are always FTAB characters and do not need to be specified.
- For SCS2, up to three FTAB characters or six hexadecimal digits can be specified. The characters NL, CR, LF, HT, and VT are always FTAB characters and do not have to be specified; however, they are received by MFS only if the Hollerith code is punched in the card if the input is from the card reader.

If no FTAB characters are defined, each device input field is considered to be of its defined length. In Record mode, when the end of a record is reached, the current field is terminated and all subsequent fields defined for that record are processed with no device data (message fill). In Stream mode, all transmissions that comprise the input message are treated as a stream of data fields unaffected by transmission boundaries. If FTABs are not defined or are not used for DPM input, each input field is considered to be of defined length except when NULL=DELETE is specified. With NULL=DELETE, if trailing nulls are encountered in a field or an entire field is null, the field is padded to defined length using the message fill character.

If FTAB characters are defined in this operand, either FORCE, MIX, or ALL can also be specified. The default is FORCE.

FORCE
Specifies that an FTAB is not required until you or a remote program enters an FTAB character. In record mode, if an FTAB is used for one field, the remaining fields of the current record must be terminated with an FTAB, regardless of length. In stream mode, if an FTAB is used for one field, the remaining fields in the message must be terminated with an FTAB.

MIX
Specifies that an FTAB is never required but can be used to terminate any input field when data is less than the defined field length.

ALL
Specifies that an FTAB must be used to terminate all fields, regardless of length, except for certain mode (MODE=) dependent conditions. In record mode, an FTAB is not required for the last field defined or entered in the record. In stream mode, an FTAB is not required for the last field defined or entered in the message.

LDEL=
Specifies two characters or four hexadecimal digits, which, if entered as the last two characters of a record of input data, cause the record to be discarded. A specification of NONE causes IMS to bypass record delete processing, except for the first record, which is always deleted if the last two characters are asterisks (**). NONE is the default for DPM devices. For other devices, the default is **.

PAGE=
Specifies output parameters as follows:

number
For printer devices, number defines the number of print lines on a printed page; for card devices, number defines the number of cards to be punched per DPAGE or physical page (if *pp* parameter is used in the DFLD statements). This value is used for validity checking. The number specified must be greater than or equal to 1 and less than 256. The default is 55.

If VTAB= is specified for SCS1 printers, then the minimum value for PAGE= is 3.

DEFN
Specifies that lines/cards are to be printed/punched as defined by DFLD statements (no lines/cards are to be removed or added to the output page).

SPACE
Specifies that each output page contains the exact number of lines/cards specified in the *number* parameter.

FLOAT
Specifies that lines/cards with no data (all blank or NULL) after formatting are to be deleted.

For 3270P and SCS1 devices, some lines having no data (that is, all blank or null) must not be deleted under the following circumstances:

- The line contains one or more set line density (SLDx=) specifications.
- A field specified as having extended attributes spans more than one line.

EJECT
Specifies that a forms eject operation should be performed for printer devices. EJECT is valid only when TYPE=FIJP, FIPB, FIFP, or SCS1. If EJECT is specified for SCS1, MFS assumes the Vertical Forms Control feature is present. The default for the sublist is BGNPP.

The sublist specifies when ejects are to be performed:

BGNPP
Specifies that an eject is to be performed before each physical page of output.

ENDPP
Specifies that an eject is to be performed after each physical page is printed.

BGNMSG
Specifies that an eject is to be performed before any data in the message is printed.

ENDMSG
Specifies that an eject is to be performed after all message data is printed.

DSCA=
Specifies a default system control area (DSCA) for output messages using this device format.
The DSCA supersedes any SCA specified in a message output descriptor if there are conflicting specifications. Normally, the functions specified in both SCAs are performed. If the DSCA= operand is specified for SCS1 or SCS2, it is ignored. If the DSCA= operand is specified for 3270P, it is ignored, except for the bit setting for "sound device alarm". If this bit is specified on the DSCA/SCA option, it is sent to the device. For TYPE=DPM-An or DPM-Bn, DSCA/SCA information is sent to a remote program or ISC subsystem only if a DFLD definition requests it.

The value specified here must be a decimal number not exceeding 65535 or X'*hhhh*'. If the number is specified, the number is internally converted to X'*hhhh*'.

The two bytes of the DSCA field should be defined as shown in the following table and Table 19 on page 203.

The following table shows the DSCA bit settings for 3270 display or SLU 2 devices or TYPE=DPM-An or DPM-Bn.

Table 17. Bit settings for DSCA field for 3270 Display, SLU 2 Devices, TYPE=DPM-An, or DPM-Bn

Byte	Bit	Setting
0	0-7	Should be 0.

Table 17. Bit settings for DSCA field for 3270 Display, SLU 2 Devices, TYPE=DPM-An, or DPM-Bn (continued)

Byte	Bit	Setting
1	0	Should be 1.
	1	Force format write (erase device buffer and write all required data).
	2	Erase unprotected fields before write.
	3	Sound device alarm.
	4	Copy output to candidate pointer. Bits 1-4 are ignored for DPM-Bn.
	5	B'0'- For 3270, protect the screen when output is sent. For DPM, demand paging can be performed. B'1'- For 3270, do not protect the screen when output is sent. For DPM-B, autopaging can be performed.
	6-7	Should be 0, except for the 3290 in partitioned format mode.

If byte 1 bit 5 is set to B'1' (unprotect screen option) for a 3275 display, and both input and output occur simultaneously (contention), the device is disconnected. For non-3275 devices, the SCA option is ignored. If byte 1 bit 5 is set to B'0', the application program can request autopaged output by setting the SCA value to B'1'. This request is honored only if present in the first segment of the first LPAGE of the output message.

If a nonzero value is specified for byte 0, or for bit 6 or 7 in byte 1, MFS overrides the specified value with zero, except for the 3290 in partitioned format mode.

For the 3290 in partitioned format mode, byte 1 bit 6 has special significance. If the DOF of the output message is the same as the DOF of the last message, then byte 1 bit 6 of the DSCA is checked for the erase/not erase partitions option before the output message is sent. The following table shows meanings of the bit 6 settings.

Table 18. 3290 partitioned format mode bit setting

Byte	Bit	Setting	Meaning
1	6	B'1'	Erase all partitions before sending output message.
		B'0111'	Do not erase existing partitions.

The default is B'0' (do not erase). If bit 6 is defined, all existing partitions are erased and the output is sent according to the specified partition paging option. If bit 6 is not defined, the output is sent according to the specified partition paging option and partitions that do not receive output remain in the state they were in before output was sent.

The following table shows the DSCA bit settings for TYPE=FIDS, FIDS3, FIDS4, FIDS7, FIJP, FIPB, or FIFP.

Table 19. Bit settings for DSCA field for TYPE=FIDS, FIDS3, FIDS4, FIDS7, FIJP, FIPB, or FIFP

Byte	Bit	Setting
0	0-7	Should be 0.
1	0	Should be 1.
	1-2	Not applicable for FIN output devices.
	3	Set 'device alarm' in output message header.
	4	Not applicable for FIN output devices.
	5-7	Should be 0.

For FIN devices, if a nonzero value is specified for byte 0, or for bits 1, 2, 5, 6, or 7 in byte 1, MFS overrides the specified value with zero.

Bits 1, 2, and 4 in byte 1 only function for 3270 and SLU 2 and are therefore not applicable to FIN. If set on, and the message is edited for an FIN output device, they are ignored.

For 3270 and FIN devices, the function specified is performed. For DPM devices, the specification is supplied to the remote program in a user-defined device field (DFLD).

FEAT=
Specifies features for this device or program group.

IGNORE
Specifies that device features are to be ignored for this device.

120 | 126 | 132
Specifies line length for 3284, and 3286 device types (TYPE=3270P).

CARD
Specifies that the device has a 3270 operator identification card reader. NOCD specifies the absence of the CARD feature.

DEKYBD
Specifies data entry keyboard feature. This feature implies PFK feature; therefore, PFK is invalid if DEKYBD is specified. NOPFK implies the absence of PFK and DEKYBD features.

PFK
Specifies that the device has program function keys. NOPFK specifies the absence of the PFK and DEKYBD features.

PEN
Specifies the selector light pen detect feature. NOPEN specifies the absence of the PEN feature.

DUAL
Specifies that the FIFP device has the dual independent forms feed feature.

132
Specifies that the FIFP device has the expanded print line feature.

1|2|3|4|5|6|7|8|9|10
Specifies customer-defined features for the SCS1, SCS2, 3270P, DPM-An, or DPM-Bn device type.

For SCS1, SCS2, and 3270P devices, FEAT= allows grouping of devices with special device characteristics. For example, FEAT=1 could group devices with a maximum of 80 print positions and no VFC, and FEAT=2 could group devices with 132 print positions and the VFC feature. FEAT=IGNORE should be specified to group together devices with a minimum set of device capabilities. For 3270P devices, when WIDTH= is specified, FEAT=(1...10) must also be specified. If FEAT=(1...10) is specified but WIDTH= is not specified, WIDTH= defaults to 120.

For DEV TYPE=DPM-An or DPM-Bn, FEAT= specifies a user-defined group of device formats so that programs with common features and dependencies can be selected together.

When IGNORE is specified, no other values should be coded in the FEAT= operand. When FEAT=IGNORE is not specified in the TERMINAL macro during system definition, the MSG statement must specify IGNORE in the SOR= operand for the device format with the IGNORE specification. Unless FEAT=IGNORE is used, FEAT= must specify exactly what was specified in the TERMINAL macro during IMS system definition. If it does not, the DFS057 error message is issued. When FEAT=IGNORE or 1-10 is specified for 3270 devices, the operands PEN=, CARD=, and PFK= can still be specified. When TYPE=3270P and FEAT=IGNORE, MFS allows a line width of 120 characters.

CARD, PFK, DEKYBD, and PEN feature values are valid only for 3270 displays. DUAL is valid only if TYPE=FIFP. If the FEAT= operand is omitted, the default features are CARD, PFK, and PEN for 3270 displays; the default line width is 120 for TYPE=3270P and 80 for TYPE=FIFP.

1, 2, 3, 4, 5, 6, 7, 8, 9, and 10 are valid values only for 3270, 3270P, 3270-An, SCS1, SCS2, DPM-An, and DPM-Bn (for DEV TYPE=). For 3270 displays, the FEAT= specifications of 1 to 5 can be used to group devices with specific features or hardware data stream dependencies.

Restriction: This keyword is optional and cannot be used with any other feature specification for 3270 displays.

When using the same format for both the 3290 and the 3180, you must specify a different value on the FEAT= operand for each device type. The FEAT parameter values selected for each device must also be specified on the TERMINAL macro in the IMS system definition.

For FIN, FIDS, FIDS3, FIDS4, FIDS7, FIJP, and FIPB, FEAT is always IGNORE. For FIFP, IGNORE is used unless 132 and DUAL are specified.

Feature operand values can be specified in any order, and only those values desired need be specified. The underlined values do not have to be specified because they are defaults. Only one value in each vertical list can be specified.

Examples: Some of the uses of the FEAT= specification are:

- `TYPE=DPM-A1,FEAT=1` could group device formats with DPAGE paging option and simulated attributes.
- `TYPE=DPM-A5,FEAT=2` could group device formats with no paging option and bit string attributes (which are not interpreted by MFS).
- `TYPE=DPM-B1,FEAT=IGNORE` could identify device formats with PPAGE paging option and a minimum set of program requirements.

PFK=

Defines an input field name to contain program function key literal or control function data (first subparameter) and, in positional or keyword format, either the literal data to be placed in the specified field, or the control function to be performed when the corresponding function key is entered (remaining subparameters).

The name of the first subparameter (the input field name that will contain the program function key literal or control function data) can be referred to by an MFLD statement and must not be used as the label of a DFLD statement within this DEV definition. The remaining subparameters can be specified in positional or keyword format. If the subparameters are in keyword format, the integer specified must be from 1 to 36, inclusive, and not duplicated. Only one PFK= operand format (positional or keyword) can be specified on a DEV statement. This operand is valid only for 3270 displays. At the time the actual format blocks are created, each literal is padded on the right with blanks to the length of the largest literal in the list. The maximum literal length is 256 bytes.

If the device supports the IMS copy function, then PFK12 invokes the copy function and the definition of PFK12 in the DEV statement is ignored; otherwise, the definition of PFK12 is honored.

If FEAT=NOPFK is specified, it is changed to PFK. The maximum number of user-defined PFKs is 36.

Control functions that can be specified are:

NEXTPP—PAGE ADVANCE
Specifies a request for the next physical page in the current output message. If no output message is in progress, no explicit response is made.

NEXTMSG—MESSAGE ADVANCE
Specifies a request to dequeue the output message in progress (if any) and to send the next output message in the queue (if any).

NEXTMSGP—MESSAGE ADVANCE PROTECT
Specifies a request to dequeue the output message in progress (if any), and send the next output message or return an information message indicating that no next message exists.

NEXTLP—NEXT LOGICAL PAGE
Specifies a request for the next logical page of the current message.

ENDMPPI—END MULTIPLE PAGE INPUT
Specifies the end of a multiple physical page input message.

PEN=

Defines an input field name to contain literal data when an immediate light pen detection of a field with a space or null designator character occurs. The literal data is defined on the DFLD statement

with the PEN= operand. (See PEN= operand on the DFLD statement.) This name can be referred to by an MFLD statement and must not be used as the label of a DFLD statement within this DEV definition. The PEN= operand is valid only for 3270 displays. If FEAT=NOPEN is specified, it is changed to PEN.

If an immediate detect occurs on a field defined with a space or null designator character, and either another field has been selected or modified or has the MOD attribute, or the PEN= operand is not defined for the DFLD, a question mark (?) is inserted in the PEN= field name.

If no immediate detection occurs or the immediate detect occurs on a field defined with an ampersand (&) designator character, the PEN= operand is padded with the fill specified in the MFLD statement.

CARD=
Defines the input field name to receive operator identification card data when that data is entered. This name can be referenced by an MFLD statement and must not be used as the label of a DFLD statement within this DEV definition. This operand is valid only if a 3270 display or SCS1 is specified. If FEAT=NOCD is specified for a 3270 display, it is changed to CARD. All control characters are removed from magnetic card input before the data is presented to the input MFLD that refers to this card field name.

For 3270 displays, an unprotected field large enough to contain the magnetic card data and control characters must be defined through a DFLD statement. Position the cursor to this field and insert the card in the reader to enter card information. The card data is logically associated with the CARD= field name, not the name used in the DFLD statement.

For device TYPE=SCS1, only card data with the operator ID (OID) character is associated with this field name. Cards with the OID character can be entered at any time during data entry. MFS treats data without the OID character as if it were data entered from the keyboard.

SYSMSG=
Specifies the label of the DFLD statements that define the device field in which IMS system messages are to be displayed. This operand is valid only if a 3270 display is specified. A DFLD with this label should be defined for each physical page within each DPAGE defined within this DEV definition. DFLDs for SYSMSG should be at least LTH=79 to prevent message truncation. The referenced DFLD can also be referenced by an MFLD statement.

FORMS=
Specifies a 1- to 16-byte literal. For the FIN, this literal is included in the output message header for each message sent to the device using this FMT. The data can be used by the FIN application program to ensure that special forms required for a given message are mounted on the device and that page size and forms alignment are established.

For SCS1 output to SLU 1 print data set components or SLU 4, this literal names the data set to receive IMS output. For 3770 programmable models defined to IMS as SLU 1 or SLU 4, however, the literal is ignored by the terminal and all print data set output goes to the SYS.INTR data set. For all SCS1 output to 3770 (nonprogrammable), SLU 1 non-PDS components or SLU 4, the literal is ignored.

For DEV TYPE DPM-An, this literal is included in the output message header. If the DPAGE or PPAGE paging option is specified, the literal is part of the special forms output message header sent as a separate transmission, followed (after a paging request from the remote program) by the DPAGE or PPAGE output message header and data records. If the default MSG option is selected, the output message header with literal is sent as the first record, followed by data records.

WIDTH=
Specifies the maximum line width for this DEV type as one of:

- Number of print positions per line of input or output data
- Number of punch positions per card of input or output data
- Card width for card reader input data

The defaults are 132 for SCS1 input and output, 80 for SCS2 input and output, and 120 for 3270P output. A specified number cannot exceed 255 for SCS1 and 249 for SCS2. Line width is specified

relative to column 1, regardless of whether a left margin value is specified in the HTAB= keyword (SCS1 and SCS2 only). The width specified must be greater than or equal to 1.

For 3270P devices, if WIDTH is specified, then FEAT=(1...10) must also be specified. If FEAT=(1...10) is specified, and WIDTH= is not specified, WIDTH= defaults to 120.

HTAB=

Specifies when TYPE=SCS1:

- Where on the device MFS should set horizontal tab stops
- Whether and when MFS should insert tab control characters in the output message to cause horizontal tabbing
- Where on the device MFS should position the left margin

If HTAB= is not specified, no horizontal tabbing is done and the left margin position is assumed to be column 1.

SET | ONLINE | OFFLINE

Specifies that MFS should set horizontal formatting controls for the device. When MFS sets horizontal format controls for the device, the following characteristics are established: maximum line width, left and right margins, and horizontal tab stops. The default is SET when the HTAB= keyword is present.

SET

Specifies that MFS should set horizontal tab stops but should not insert tab control characters into the output message. You can then use horizontal tabbing on subsequent input.

ONLINE

Specifies that MFS should set horizontal tab stops at the specified (HT=) locations and insert tab control characters during online processing.

OFFLINE

Specifies that MFS should set horizontal tab stops at the specified (HT=) locations and insert tab control characters during offline compilation of the format.

1 | lm (left margin)

Specifies the column position of the left margin. The default is 1. The value specified must be less than the WIDTH= value.

HT=

Specifies from 1 to 10 horizontal tab stop locations. The values specified must be relative to position 1, equal to or greater than the left margin value, and less than the WIDTH= value.

VT=

Specifies that MFS should insert tab control characters at the specified locations. From 1 to 11 vertical tab stop locations can be specified. If VTAB= is specified, the VT= values specified must be relative to line 1 and equal to or less than the bottom margin specified on the VTAB= keyword. If VTAB= is not specified, the VT= values must be equal to or less than the page depth specified in the PAGE= keyword. The maximum value is 255. If a value greater than 255 is specified, 255 is assumed and no error message is generated. VT= is valid only when TYPE=SCS1. If PAGE=(n,FLOAT) is specified, VT= is invalid.

X'00' is accepted as a valid tab stop only if VTAB= is also specified.

Together with VTAB= and PAGE=, VT= comprises a data stream to set the vertical format of the page. *tm* on the VTAB= keyword must be greater than or equal to 1 and less than *t1* on the VT= keyword. *bm* on the VTAB= keyword must be greater than or equal to *t11* on the VT= keyword and less than or equal to the maximum page length specified on the PAGE= keyword.

VTAB=

For SCS1 printers, specifies top (*tm*) and bottom (*bm*) page margins. Together with VT= and PAGE=, VTAB= comprises a data stream to set the vertical format of the page. *tm* must be greater than or equal to 1 and less than *t1* on the VT= keyword. The maximum *tm* is 253.

bm must be greater than or equal to *t11* on the VT= keyword and less than or equal to the maximum page length specified on the PAGE= keyword. *bm* must be at least two greater than *tm*. If VTAB= is specified, then the PAGE= value must be 3 or greater.

A form feed (FF) is inserted after the set vertical format (SVF) data stream if the top margin (*tm*) specified on the VTAB= keyword is not equal to 1.

If PAGE=(n,FLOAT) is specified, VTAB= is invalid.

SLDI=
For SCS1 printers, specifies the line density for an output message in lines per inch. (See also SLDP=). SLDI= can also be specified on the DFLD statement. The SLDI= value must be from 1 through 72 and consistent with the architecture of the device for which it is specified (see the appropriate device or component manual).

If SLDI= is specified both on the DEV statement and the DFLD statement, two SLD data streams are created. One is sent at the beginning of a message to set the line density. The second is sent within the message, just prior to the field on which the SLDI= specification is encountered, but after any vertical tabs and new line characters.

Restriction: You cannot specify both SLDI= and SLDP= on the DEV statement.

The SLDI= specification within the message changes the line density from that set at the beginning of the message, and this latter line density remains in effect until explicitly reset.

SLDP=
For SCS1 printers, specifies the line density for an output message in points per inch. (See also SLDI=). SLDP= can also be specified on the DFLD statement. The SLDP= value must be from 1 through 72 and consistent with the architecture of the device for which it is specified (see the appropriate device or component manual).

If SLDP= is specified both on the DEV statement and the DFLD statement, two SLD data streams are created. One is sent at the beginning of a message to set the line density. The second is sent within the message, just prior to the field on which the SLDP= specification is encountered, but after any vertical tabs and new line characters.

Restriction: You cannot specify both SLDP= and SLDI= on the DEV statement.

The SLDP= specification within the message changes the line density from that set at the beginning of the message, and this latter line density remains in effect until explicitly reset.

Tip: When you define set line density (SLDx) keywords, ensure that forms alignment is maintained. If SLDx= is improperly defined, loss of forms alignment can occur.

VERSID=
Specifies any two-character or 2-byte hexadecimal value as the version ID. If MFS is specified or if the VERSID keyword is not specified, MFS calculates the version ID. MFS is the default.

The version ID is calculated by MFS and is based on the date and time stamp that an FMT definition has compiled. The value is printed on the MFS Language utility output so you can refer to it in format definitions.

SUB=
Specifies the character used by MFS to replace any X'3F' characters in the input data stream. No translation occurs if this parameter is specified as X'3F' or this parameter is not specified, or the input received bypasses MFS editing. The specified SUB character should not appear elsewhere in the data stream; therefore, it should be nongraphic.

X'*hh*'
Character whose hexadecimal representation is '*hh*' replaces all X'3F' in the input data stream.

C'*c*'
Character '*c*' replaces all X'3F' in the input data stream.

PDB=

(For the 3290 or 3180 in partitioned format mode) specifies the name of the Partition Descriptor Block that is used to describe the partition set for an output or input message. This parameter is valid only for DEV statements that specify TYPE=3270-An.

Related concepts

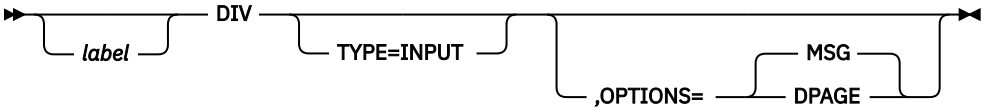
MFS message formats (Application Programming APIs)

DIV statement

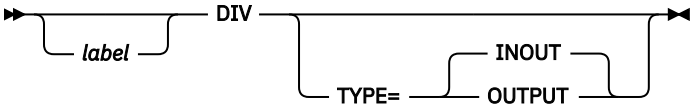
The DIV statement defines device formats within a DIF or DOF. The formats are identified as input, output, or both input and output, and can consist of multiple physical pages.

For DEV TYPE=SCS1, SCS2, or DPM-AN, two DIV statements can be defined: DIV TYPE=OUTPUT and DIV TYPE=INPUT. For all other device types, only one DIV statement per DEV is allowed.

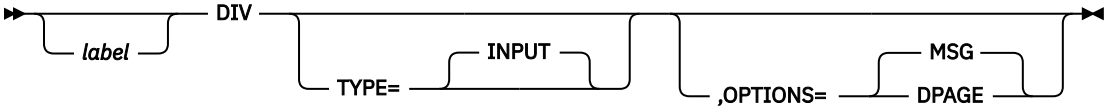
Format for DEV TYPE=SCS1, or SCS2 and DIV TYPE=INPUT



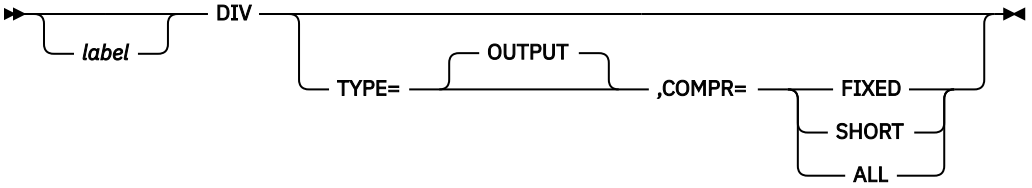
Format for DEV TYPE=3270 or 3270-An



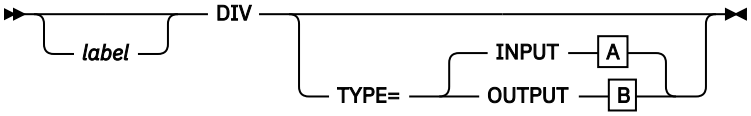
Format for DEV TYPE=FIN



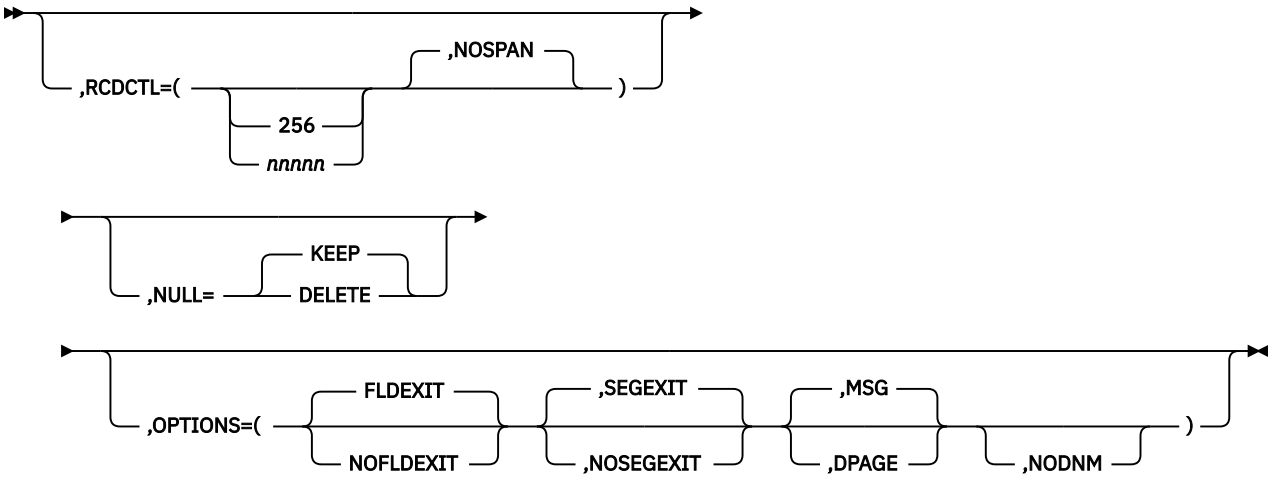
Format for DEV TYPE=SCS1, SCS2, 3270P, FIDS, FIDS3, FIDS4, FIDS7, FIJP, FIPB, or FIFP and DIV TYPE=OUTPUT



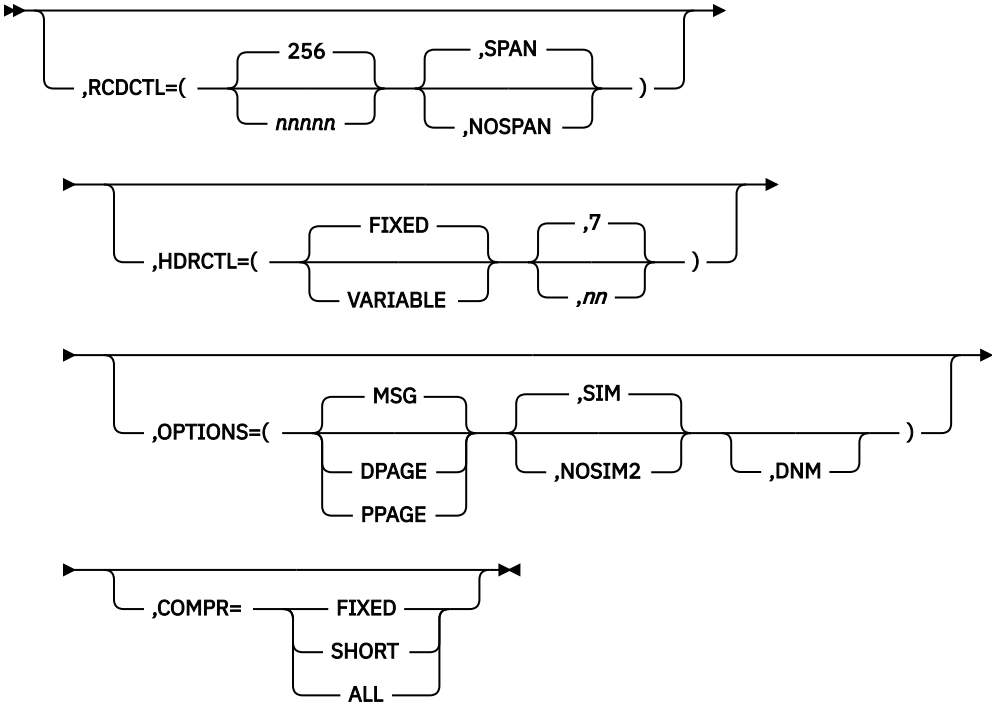
Format for DEV TYPE=DPM-An



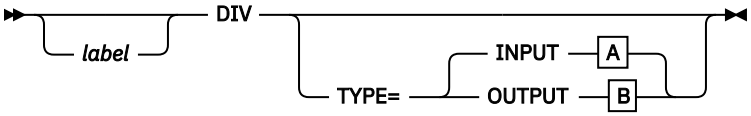
A



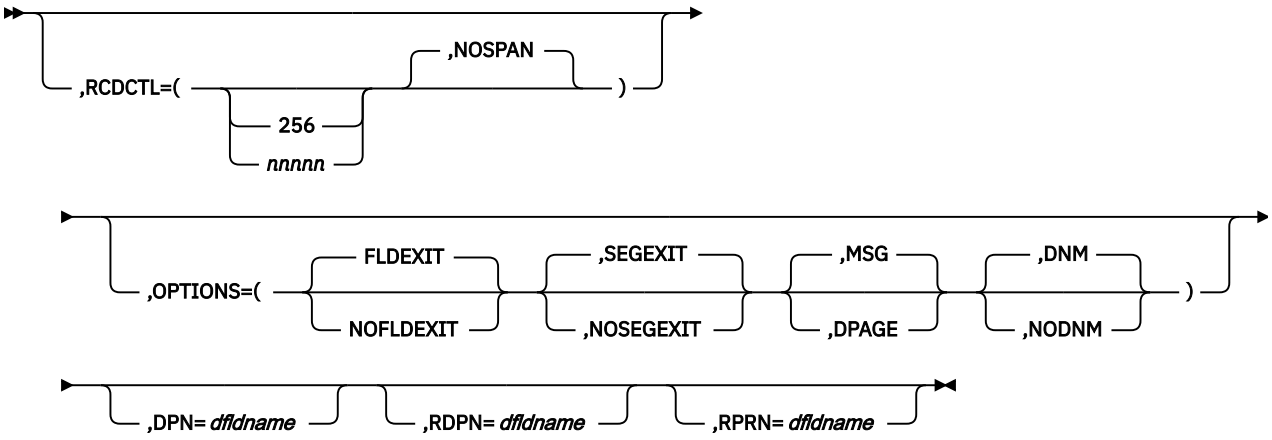
B



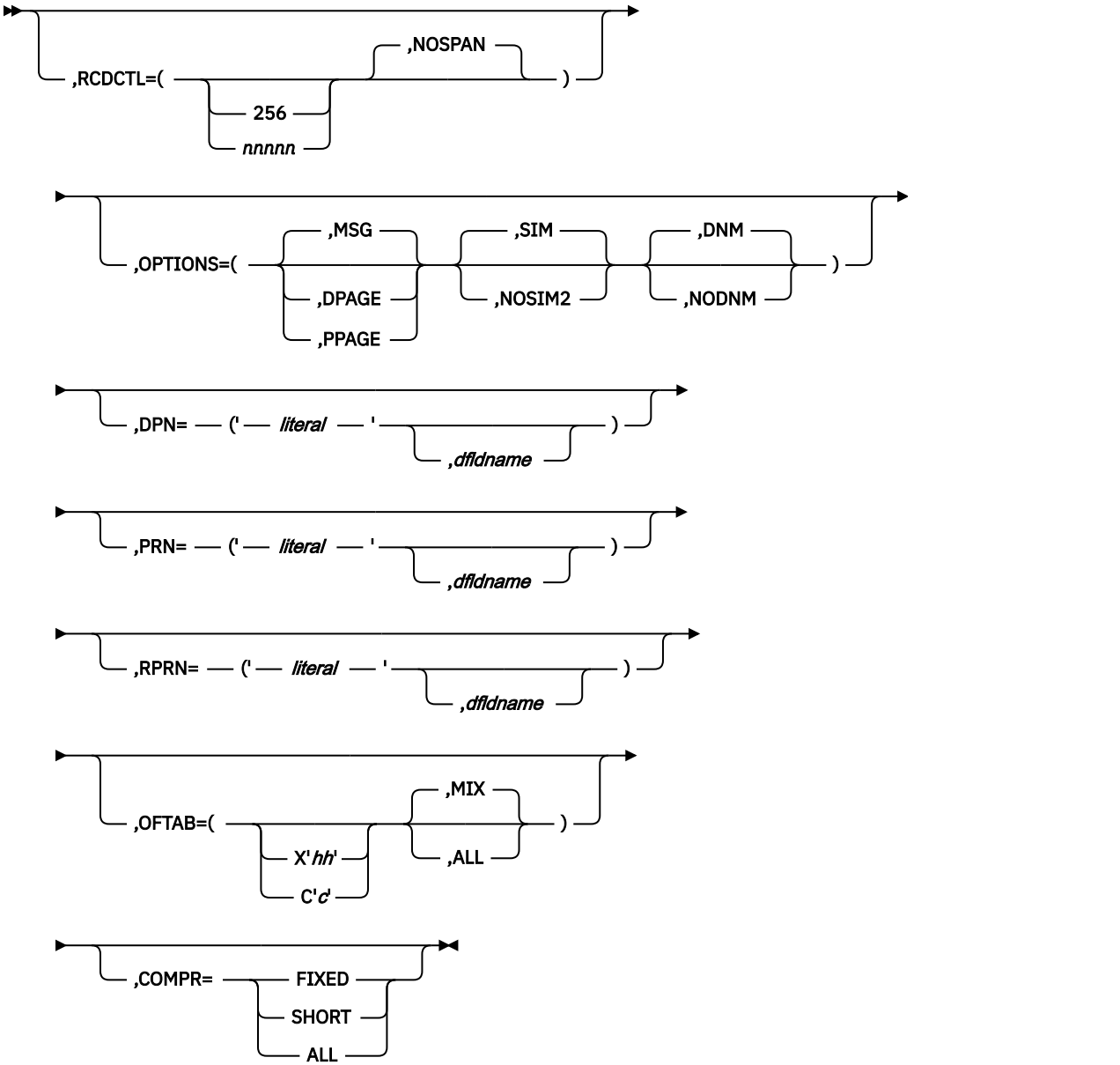
Format for DEV TYPE=DPM-Bn



A


,RCDCTL=(
256
nnnnn
,NOSPAN
)
,OPTIONS=(
FLDEXIT
NOFLDEXIT
,SEGEXIT
,NOSEGEXIT
,MSG
,DPAGE
,DNM
,NODNM
)
,DPN= dfldname
,RDPN= dfldname
,RPRN= dfldname

B


,RCDCTL=(
256
nnnnn
,NOSPAN
)
,OPTIONS=(
,MSG
,DPAGE
,PPAGE
,SIM
,NOSIM2
,DNM
,NODNM
)
,DPN= ('literal'
,dfldname
)
,PRN= ('literal'
,dfldname
)
,RPRN= ('literal'
,dfldname
)
,OFTAB=(
X'hh'
C'c'
,MIX
,ALL
)
,COMPR=
FIXED
SHORT
ALL

Parameters

label
A one- to eight-character alphanumeric name can be specified to uniquely identify this statement.

TYPE=
Describes an input only format (INPUT), an output only format (OUTPUT), or both (INOUT).

If DIV TYPE=OUTPUT or TYPE=INPUT is specified, certain DEV statement keywords are applicable.

For example, specifying WIDTH=80 for DEV TYPE=SCS1 indicates that fields can be printed in columns 1 through 80 on output and received from columns 1 through 80 on input. Specifying WIDTH=80 for DEV TYPE=SCS2 indicates that both the card reader and card punch have the same number of punch positions. Specifying WIDTH=80 and HTAB=(SET,5) for DEV TYPE=SCS1 indicates that fields can be printed in columns 5 through 80 and received from columns 5 through 80 on input. In this case DFLD POS=(1,5) or POS=5 on input is the same as if you specified column 1 and a left margin position at 1. You enter data the same way, regardless of where the left margin is currently set.

RCDCTL=
This parameter is valid only if MODE=RECORD is specified on the DEV statement. For DEV TYPE=DPM-An or DPM-Bn only, RCDCTL specifies the maximum length of an input or output transmission record. For DPM-An, RCDCTL specifies whether (SPAN) or not (NOSPAN) fields can span records. The RCDCTL number cannot be larger than 32000 and should not be less than the length of the message output header (For DPM-An, see HDRCTL discussion.) The default value is 256. RCDCTL creates record definitions even if RCD statements are used in the same format definition.

- For DEV TYPE=DPM-An or TYPE=Bn and DIV TYPE=INPUT

  For an input format definition, fields must not span record boundaries, and therefore must be within the length specified by the RCDCTL value. NOSPAN is the default.

- For DEV TYPE=DPM-An or Bn and DIV TYPE=OUTPUT

  The RCDCTL size specified should be less than or equal to the output buffer size specified in the OUTBUF= macro at IMS system definition. If the RCDCTL size is greater than the OUTBUF value specified, one record might require multiple output transmissions and might produce undesirable results in the remote program. If fields do not exactly fit in the defined records, and NOSPAN has been specified, records might not be completely filled.

  The RCDCTL also specifies whether (SPAN) (for DPM-An only) or not (NOSPAN) a field can span record boundaries. If SPAN is specified (for DPM-An only), some fields can span a record boundary (but never a PPAGE boundary), and the remote program must include logic to associate the partial fields or deal with them separately.

  If NOSPAN is specified, every field is entirely contained within a record and no field will have a length greater than the RCDCTL value specified.

  The first data field is the first field of the message for OPTIONS=MSG. The first data field is the first field of the DPAGE or PPAGE for OPTIONS=DPAGE and PPAGE respectively. If the first data field does not fit in the same record as the output message header, and if OPTIONS=DPAGE or PPAGE has been specified, the first data record will be sent in the next transmission. The output message header will be transmitted by itself (as is always the case for OPTIONS=MSG).

NULL=
For DEV TYPE=DPM-An and DIV TYPE=INPUT only, NULL= specifies whether MFS is to ignore (KEEP) or search for and replace (DELETE) trailing nulls in fields. If NULL=DELETE is specified, MFS searches input message fields for trailing nulls or for fields that are all nulls, and replaces the nulls with the fill character specified in the message definition.

OPTIONS=
For DIV TYPE=INPUT, the OPTIONS keyword specifies the exit routines to be called, the type of paging or delivery requested, and, for DPM-Bn only, the selection of the DPAGE data name to be used to map data.

For DIV TYPE=OUTPUT, the OPTIONS= keyword specifies the type of paging or delivery requested, the type of attribute processing requested, and, for DPM-Bn only, the selection of the DPAGE data name to be used to map data.

For DPM output messages, the option selection determines how records are constructed for transmission to the remote program or ISC subsystem and effects the distribution of processing and logic between the IMS application program and the remote program or ISC subsystem.

- For DEV TYPE=DPM-An or TYPE=DPM-Bn and DIV TYPE=INPUT

  FLDEXIT|NOFLDEXIT
  SEGEXIT|NOSEGEXIT
  Input data from this device type can be partially edited by the remote program before it is sent to IMS. For input format definitions, this parameter specifies whether (FLDEXIT and SEGEXIT) or not (NOFLDEXIT and NOSEGEXIT) exit routines specified in the MSG definition MFLD and SEG statements, respectively, are to be called for this DPM format. If NOFLDEXIT or NOSEGEXIT is specified, the corresponding exit routine is bypassed. FLDEXIT and SEGEXIT are the defaults.

  MSG
  Specifies that an input message can be created from a single DPAGE.

  DPAGE
  Specifies that an input message can be created from multiple DPAGEs. If multiple DPAGE input is not requested in MFS definitions, messages can not be created from more than one DPAGE. In this case:

  If a single DPAGE is transmitted and contains more data than defined for the DPAGE selected, the input message is rejected and an error message is issued.

  If multiple DPAGEs are transmitted, the input message is rejected and an error message is issued.

  NODNM (DPM-An only)
  DNM/NODNM (DPM-Bn only)
  When a data name (DNM) is specified or defaulted to (DPM-Bn only), a specific DPAGE is selected to map the current or only data transmission when:

  The DPAGE data name is supplied as the DSN parameter in the message header, and

  The DPAGE data name matches a defined DPAGE data name.

  If these conditions are not met, the last defined DPAGE name is used to map the data, unless the DPAGE is defined as conditional.

  When no data name (NODNM) is specified (for either DPM-An or -Bn) MFS selects a specific DPAGE by performing a conditional test on the data received and the COND= parameter.

- For DEV TYPE=SCS1, SCS2, FIN, and DIV TYPE=INPUT

  MSG
  Specifies that an input message can be created from a single DPAGE.

  DPAGE
  Specifies that an input message can be created from multiple DPAGEs. If multiple DPAGE input is not requested in MFS definitions, messages can *not* be created from more than one DPAGE. In this case:

  – If a single DPAGE is transmitted and contains more data than defined for the DPAGE selected, the input message is rejected and an error message is sent to the other subsystem.
  – If multiple DPAGEs are transmitted, the input message is rejected and an error message is sent to the other subsystem.

- For DEV TYPE=DPM-An or TYPE=DPM-Bn and DIV TYPE=OUTPUT

MSG
Is the default and specifies that IMS will transmit all the DFLDs within a message together as a single message group. The message is preceded by an output message header. All DFLDs are transmitted. For DPM-Bn, the data structure name is optional in the header.

DPAGE
Specifies that IMS will transmit all DFLDs that are grouped in one logical page together. The logical page will be transmitted in one or more records. If PPAGE statements are defined with the DPAGE, each PPAGE statement begins a new record. An additional logical page will be sent when a paging request is received from the remote program. Each logical page is preceded by an output message header, and the label on the DPAGE is placed in the header. For DPM-Bn, the data structure name is optional in the DD header and depends on the specification of DNM/NODNM.

PPAGE
Specifies that IMS will transmit the DFLDs that are grouped in one presentation page (PPAGE) together in one chain. The presentation page will be transmitted in a group of one or more records. An additional presentation page will be sent when a paging request is sent to IMS from the remote program. Each presentation page is preceded by an output message header, and the label on the PPAGE statement is placed in the header. For DPM-Bn, the data structure name is optional in the DD header and depends on the specification of DNM/NODNM.

SIM/NOSIM2
Specifies whether (SIM) or not (NOSIM2) MFS is to simulate attributes. SIM, the default, indicates that MFS is to simulate the attributes specified by the IMS application program and place the simulated attributes in corresponding DFLDs that are defined with ATTR=YES or YES,nn. The first byte of the field is used for the simulated attributes. If the MFLD does not supply 3270 attribute information (by means of the ATTR=YES or YES,nn operand) for the corresponding DFLD specifying ATTR=YES or YES,nn, a blank is sent in the first byte of the field. The application designer of the remote program or ISC subsystem is responsible for interpreting the simulated attribute within the remote program or ISC subsystem.

If NOSIM2 is specified, MFS sends a 2-byte bit string to the remote program or subsystem. This bit string is sent exactly as received from the IMS application program. 3270 extended bytes, if any (ATTR=YES,nn), are always sent as received from the application program and follow the 2-byte string of 3270 attributes. If the MFLD does not supply attribute information, binary zeros are sent in the two bytes preceding the data for the field.

See ATTR= on the DFLD statement for additional information.

DNM (DPM-An only)
Can be used with the FORMS= keyword on the DEV statement to specify a literal in the message header. This parameter is optional.

DNM/NODNM (DPM-Bn only)
If DNM is specified or defaulted to, MFS includes the following in the DD header:

- The FMT name, if OPTIONS=MSG
- The DPAGE name, if OPTIONS=DPAGE
- The PPAGE name, if OPTIONS=PPAGE

If NODNM is specified, no data structure name (DSN) is supplied in the DD header.

HDRCTL=
Specifies, for DEV TYPE=DPM-An and DIV TYPE=OUTPUT only, the characteristics of the output message header.

FIXED
Specifies that a fully padded output message header is to be sent to the remote program. The structure of the fixed output message header is the same for all DPM output messages built using this FMT definition. The base DPM output message header has a length of 7, and includes the version ID.

VARIABLE
Specifies that MIDNAME and DATANAME will have trailing blanks omitted and their length fields adjusted accordingly. If MIDNAME is not used, neither the MIDNAME field nor its length is present.

nn
Specifies the minimum length of the header, that is, the base header without MFS fields. The default is 7, which is the length of the base message header for DPM. Specifying other than 7 might cause erroneous results in the remote program.

The parameters referenced as RDPN=, DPN=, PRN=, and RPRN= refer to both the ISC ATTACH function management header and the equivalent ISC SCHEDULER function management header.

RDPN=
For DIV TYPE=INPUT, the dfldname specification permits the suggested return destination process name (RDPN) to be supplied in the input message MFLD referencing this dfldname. If dfldname is not specified, no RDPN is supplied in the input message.

DPN=
For DIV TYPE=OUTPUT, the 'literal' specification requests MFS to use this literal as the DPN in the output ATTACH message header. The literal cannot exceed 8 characters. If the dfldname is also specified, the data supplied in the MFLD referencing this dfldname is used as the DPN in the output ATTACH message header. If no output message MFLD reference to the dfldname exists, the 'literal' is used. If the data in the MFLD referencing the dfldname is greater than 8 characters, the first 8 characters are used.

PRN=
For DIV TYPE=INPUT, the dfldname specification permits the suggested primary resource name (PRN) to be supplied in the input message MFLD referencing this dfldname. If the dfldname is not specified, no PRN is supplied in the input message to the application program.

For DIV TYPE=OUTPUT, the 'literal' specification requests MFS to use this literal as the PRN in the output ATTACH message header. The literal cannot exceed 8 characters. If the dfldname is also specified, the data supplied in the MFLD referencing this dfldname is used as the PRN in the output ATTACH message header. If no output message MFLD reference to the dfldname exists, the 'literal' is used. If the data in the MFLD referencing the dfldname is greater than 8 characters, the first 8 characters are used.

RPRN=
For DIV TYPE=INPUT, the dfldname specification permits the suggested return primary resource name (RPRN) to be supplied in the input message MFLD referencing this dfldname. If dfldname is not specified, no RPRN is supplied in the input message to the application program.

For DIV TYPE=OUTPUT, the 'literal' specification requests MFS to use this literal as the suggested return primary resource name (RPRN) in the output ATTACH message header. The literal cannot exceed 8 characters. If the dfldname is also specified, the data supplied in the MFLD referencing this dfldname is used as the RPRN in the output ATTACH message header. If no output message MFLD reference to the dfldname exists, the 'literal' is used. If the data in the MFLD referencing the dfldname is greater than 8 characters, the first 8 characters are used.

OFTAB=
Directs MFS to insert output field tab separator characters in the output data stream for the message. If OPTIONS=DNM and OFTAB, then the OFTAB character is placed in the DD header and an indicator is set to MIX or ALL. If OPTIONS=NODNM, then no DD header is sent.

X'*hh*'
Character whose hexadecimal representation is "hh" is used as the output field tab separator character. Specification of X'3F' or X'40' is invalid.

C'c'
Character "c" is used as the output field tab separator character. Specification of C""b is invalid.

Restriction: The character specified cannot be present in the data stream from the IMS application program. If it is present, it is changed to a blank (X'40').

If an output field tab separator character is defined, either MIX or ALL can also be specified. Default value is MIX.

MIX
Specifies that the output field tab separator character is to be inserted into each individual field with no data or with less data than the defined DFLD length.

ALL
Specifies that the output field tab separator character is to be inserted into all fields, regardless of data length.

COMPR=
Requests MFS to remove trailing blanks from short fields, fixed-length fields, or all fields presented by the application program.

For DPM-AN devices, trailing blanks are removed at the end of a segment if all of the following conditions are true:

1. FILL=NULL or FILL=PT is specified.
2. GRAPHIC=YES is specified for the current segment being mapped.
3. OPT=1 or OPT=2 is specified in the MSG segment.

If conditions 1, 2, and 3 are met, replacement of trailing blanks occurs as follows:

FIXED
Specifies that trailing blanks from fixed-length fields are to be replaced by nulls.

SHORT
Specifies that trailing blanks fields shortened by the application program are to be replaced by nulls.

ALL
Specifies that trailing blanks from all fields are to replaced by nulls.

The trailing nulls are then compressed at the end of the record. See the description of the FILL= operand for additional information.

For DPM-BN devices, trailing blanks are removed if all of the following conditions are true:

1. OFTAB is specified on the current DIV statement, or FILL=NULL or FILL=PT is specified.
2. GRAPHIC=YES is specified for the current segment being mapped.
3. OPT=1 or OPT=2 is specified in the MSG segment.

If conditions 1, 2, and 3 are met, the removal of trailing blanks occurs as follows:

FIXED
Specifies that trailing blanks are to be removed from fixed-length fields.

SHORT
Specifies that trailing blanks are to be removed from fields shortened by the application program.

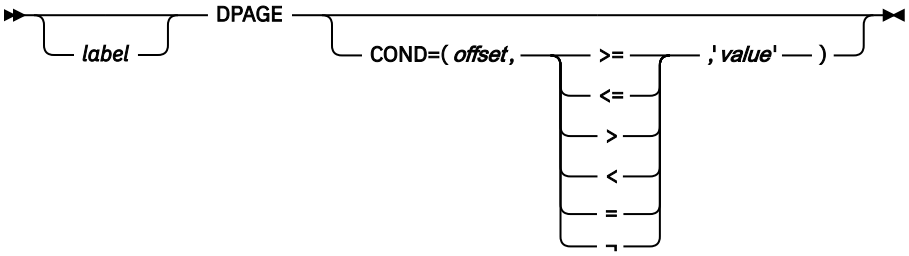
ALL
Specifies that trailing blanks are to be removed from all fields.

DPAGE statement

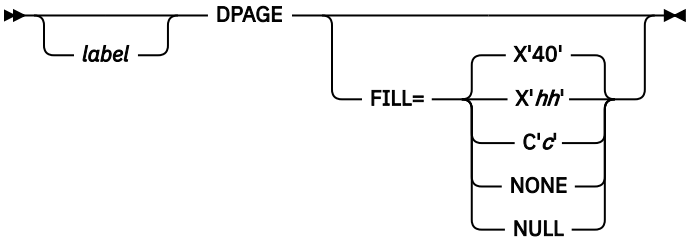
The DPAGE statement defines a logical page of a device format.

This statement can be omitted if none of the message descriptors referring to this device format (FMT) contains LPAGE statements and if no specific device option is required.

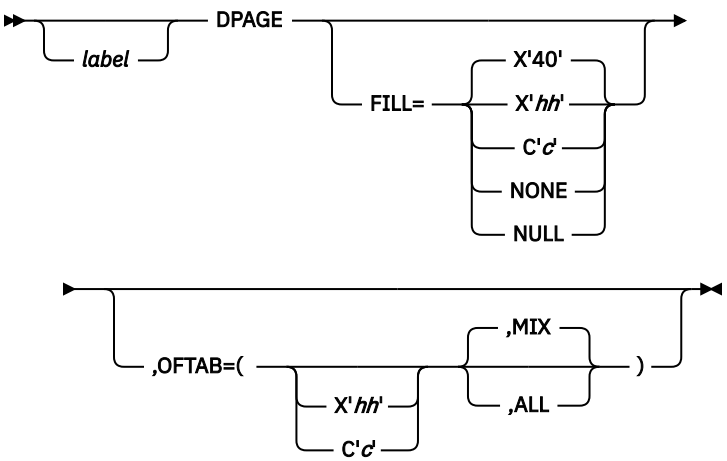
Format for DEV TYPE=DPM-An, or DPM-Bn and DIV TYPE=INPUT



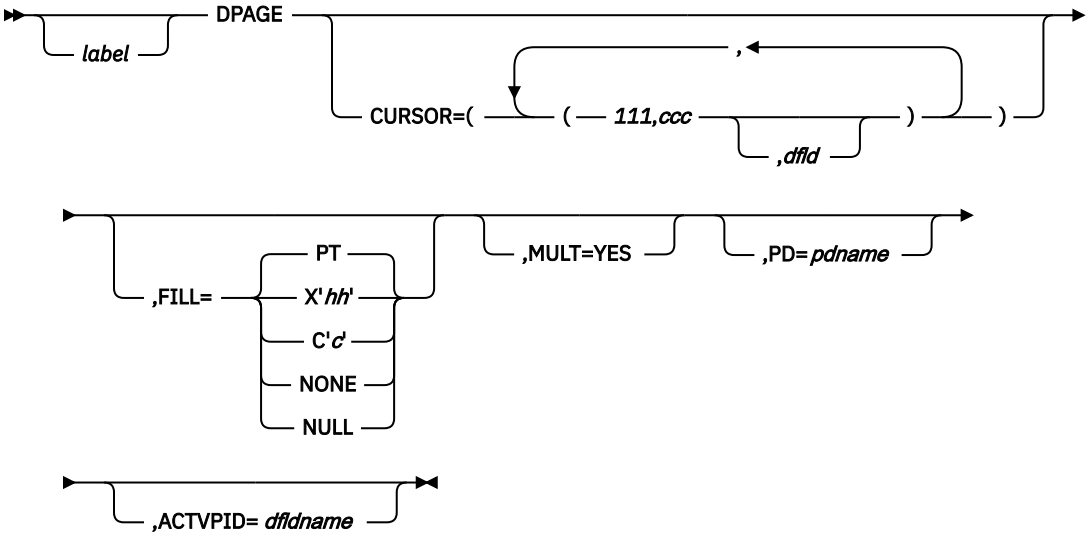
Format for DEV TYPE=DPM-An and DIV TYPE=OUTPUT



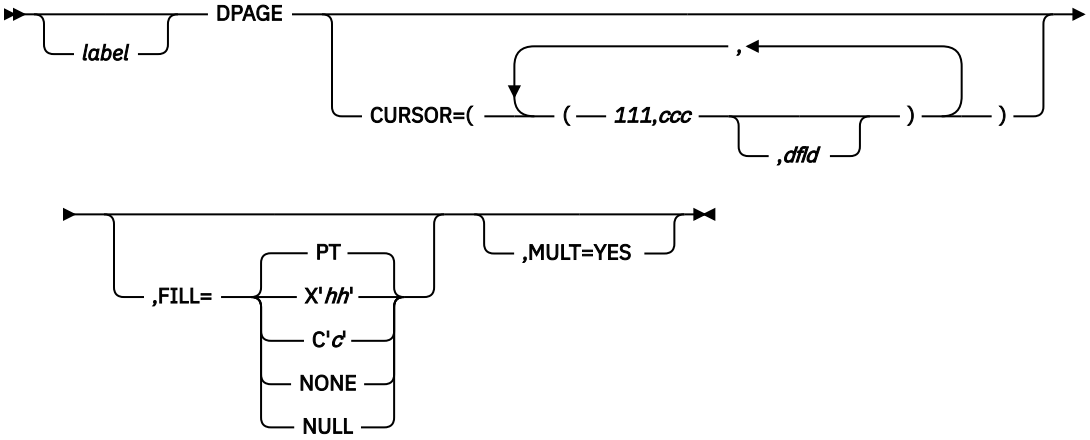
Format for DEV TYPE=DPM-Bn and DIV TYPE=OUTPUT



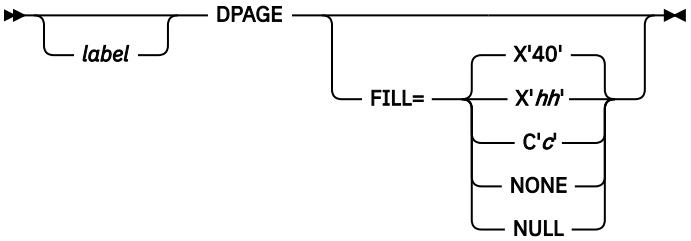
Format for DEV TYPE=3270-An



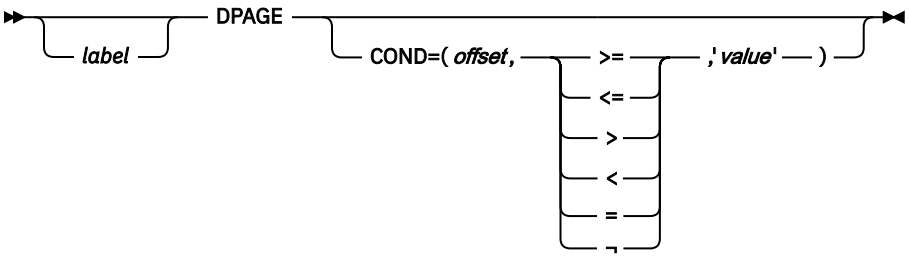
Format for DEV TYPE=3270



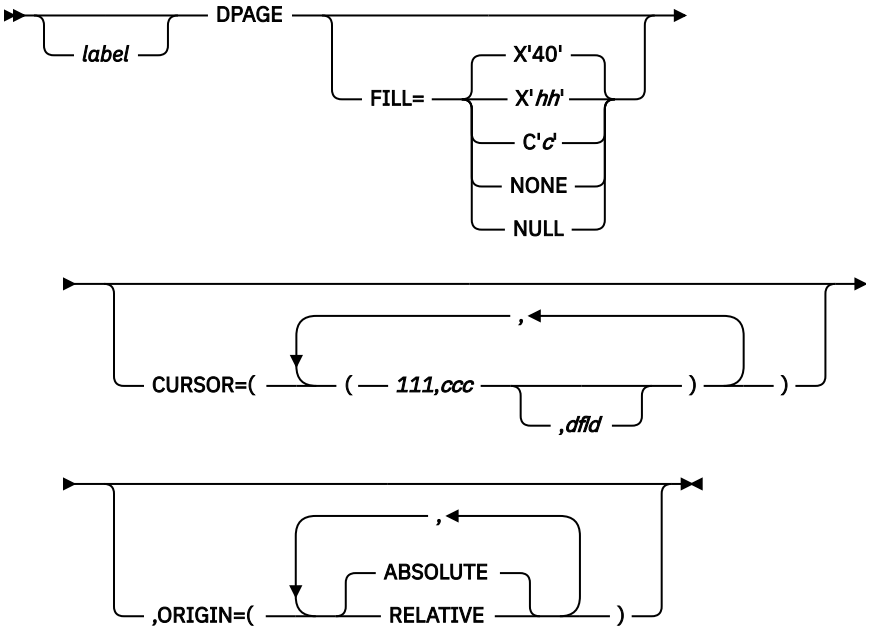
Format for DEV TYPE=3270P



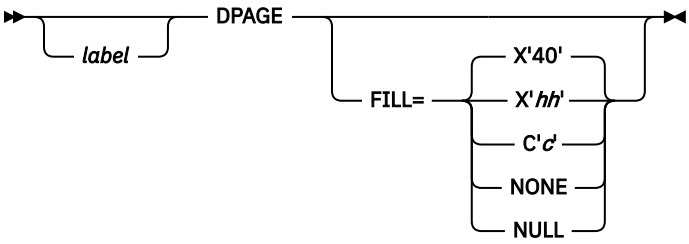
Format for DEV TYPE=FIN



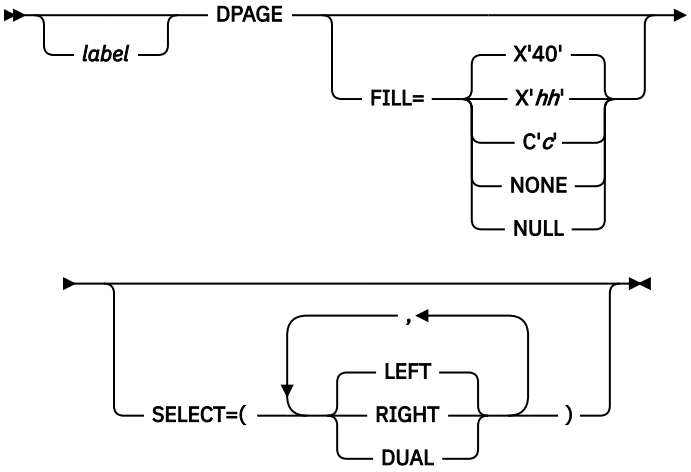
Format for DEV TYPE=FIDS, FIDS3, FIDS4, or FIDS7



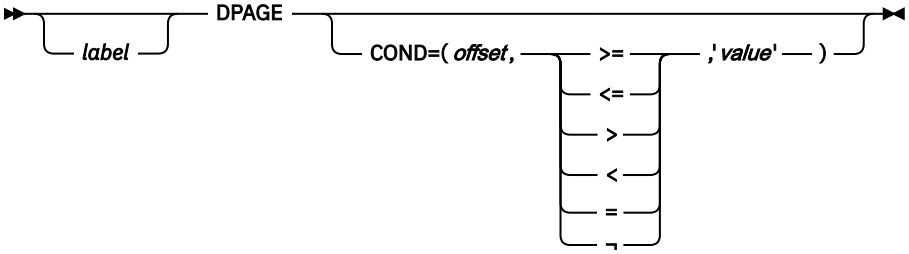
Format for DEV TYPE=FIJP or FIPB



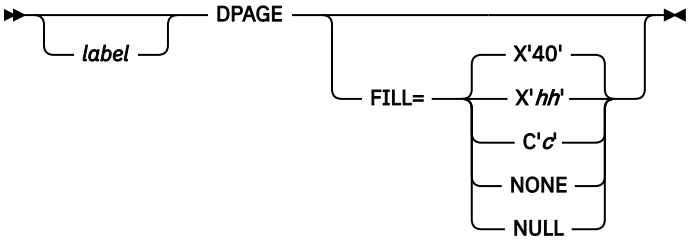
Format for DEV TYPE=FIFP



Format for DEV TYPE=SCS1 or SCS2 and DIV TYPE=INPUT



Format for DEV TYPE=SCS1 or SCS2 and DIV TYPE=OUTPUT



Parameters

label

A 1- to 8-byte alphanumeric name can be specified for this device format that contains LPAGE SOR= references, or if only one DPAGE statement is defined for the device. If multiple DEV statements are defined in the same FMT definition, each must contain DPAGE statements with the same label.

For device type DPM-An and DIV statement OPTIONS=DPAGE, this name is sent to the remote program as the data name in the output message header. If the label is omitted, MFS generates a diagnostic name and sends it to the remote program in the header. If the DPAGE statement is omitted, the label on the FMT statement is sent in the output message header. If OPTIONS=DNM, the label on the FMT statement is sent as the DSN in the DD header.

COND=

Specifies a conditional test to be performed on the first input record. The offset specified is relative to zero. The specification of the offset must allow for the LLZZ field of the input record (for example, the first data byte is at offset 4). If the condition is satisfied, the DFLDs defined following this DPAGE will be used to format the input. When no conditions are satisfied, the last defined DPAGE will be used

only if the last defined DPAGE does not specify COND=. If the COND= parameter is specified for the last DPAGE defined and the last defined DPAGE condition is not satisfied, the input message will be rejected. Multiple LPAGE definitions are allowed in message input definitions.

If this keyword is specified, and OPTIONS=NODNM is specified on the DIV statement, this specification is used for DPAGE selection. If this keyword is specified and OPTIONS=DNM is specified on the DIV statement, the COND= specification is ignored and the data structure name from the DD header is used for DPAGE selection.

Lowercase data entered from Finance, SCS1, or SCS2 keyboards is not translated to uppercase when the COND= comparison is made. Therefore, the literal operand must also be in lowercase.

FILL=
Specifies a fill character for output device fields. Default value for all device types except the 3270 display is X'40'; default for the 3270 display is PT. For 3270 output when EGCS fields are present, only FILL=PT or FILL=NULL should be specified. A FILL=PT erases an output field (either a 1- or 2-byte field) only when data is sent to the field, and thus does not erase the DFLD if the application program message omits the MFLD. For DPM-Bn, if OFTAB is specified, FILL= is ignored and FILL=NULL is assumed.

NONE
Must be specified if the fill character from the message output descriptor is to be used to fill the device fields.

X'*hh*'
Character whose hexadecimal representation is 'hh' will be used to fill the device fields.

C'*c*'
Character '*c*' will be used to fill the device fields.

NULL
Specifies that fields are not to be filled. For devices other than the 3270 display, 'compacted lines' are produced when message data does not fill the device fields.

For DPM-An devices, trailing nulls (X'3F') are removed from all records transmitted to the remote program or subsystem. Trailing nulls are removed up to the first non-null character. Null characters between non-null characters are transmitted. If the entire record is null, but more data records follow, a record containing a single null is transmitted to the remote program. If the entire record is null and more records follow, if OPTIONS=MSG or DPAGE, or in a PPAGE, if OPTIONS=PPAGE, then all null records are deleted to the end of that DPAGE or PPAGE.

PT
Is identical to NULL except for the 3270 display. For the 3270 display, specifies that output fields that do not fill the device field (DFLD) are followed by a program tab character to erase data previously in the field; otherwise, this operation is identical to FILL=NULL.

For 3270 display devices, any specification with a value less than X'3F' is changed to X'00' for control characters or to X'40' for other nongraphic characters. For all other devices, any FILL=X'hh' or FILL=C'c' specification with a value less than X'3F' is ignored and defaulted to X'3F' (which is equivalent to a specification of FILL=NULL).

MULT=YES
Specifies that multiple physical page input messages will be allowed for this DPAGE.

CURSOR=
Specifies the position of the cursor on a physical page. Multiple cursor positions might be required if a logical page or message consists of multiple physical pages. The value lll specifies line number, ccc specifies column; both lll and ccc must be greater than or equal to 1. The cursor position must either be on a defined field or defaulted. The default lll,ccc value for 3270 displays is 1,2. For Finance display components, if no cursor position is specified, MFS will not position the cursor—the cursor is normally placed at the end of the output data on the device. For Finance display components, all cursor positioning is absolute, regardless of the ORIGIN= parameter specified.

The dfld parameter provides a method for supplying the application program with cursor information on input and allowing the application program to specify cursor position on output.

Tip: Use the cursor attribute facility (specify ATTR=YES in the MFLD statement) for output cursor positioning.

The dfld parameter specifies the name of a field containing the cursor position. This name can be referenced by an MFLD statement and must not be used as the label of a DFLD statement in this DEV definition. The format of this field is two binary halfwords containing line and column number, respectively. When this field is referred to by a message input descriptor, it will contain the cursor position at message entry. If referred to by a message output descriptor, the application program places the desired cursor position into this field as two binary halfwords containing line and column, respectively. Binary zeros in the named field cause the specified lll,ccc to be used for cursor positioning during output. During input, binary zeros in this field indicate that the cursor position is not defined. The input MFLD referring to this dfld should be defined within a segment with GRAPHIC=NO specified or should use EXIT=(0,2) to convert the binary numbers to decimal.

ORIGIN=
Specifies page positioning on the Finance display for each physical page defined. Default value is ABSOLUTE.

ABSOLUTE
Erases the previous screen and positions the page at line 1 column 1. The line and column specified in the DFLD statement will become the actual line and column of the data on the screen.

RELATIVE
Positions the page starting on column 1 of the line following the line where the cursor is positioned at time of output. Results might be undesirable unless all output to the device is planned in a consistent manner.

OFTAB=
Directs MFS to insert the output field tab separator character specified on this DPAGE statement for the output data stream of the DPAGE being described.

X'*hh*'
Character whose hexadecimal representation is 'hh' is used as the output field tab separator character. Specification of X'3F' or X'40' is invalid.

C'*c*'
Character '*c*' is used as the output field tab separator character. Specification of C' ' is invalid.

Restriction: The character specified cannot be present in data streams from the IMS application program. If it is present, it is changed to a blank (X'40').

If the output field tab separator character is defined, either MIX or ALL can also be specified. Default value is MIX.

MIX
Specifies that an output field tab separator character is to be inserted into each individual field with no data or with data less than the defined DFLD length.

ALL
Specifies that an output field tab separator character is to be inserted into all fields, regardless of data length.

SELECT=
Specifies carriage selection for a FIFP device with FEAT=DUAL specified in the previous DEV statement. It is your responsibility to ensure that proper forms are mounted and that left margins are set properly. Default value is LEFT.

LEFT
Causes the corresponding physical page defined in this DPAGE to be directed to the left platen.

RIGHT
Causes the corresponding physical page defined in this DPAGE to be directed to the right platen.

DUAL
Causes the corresponding physical page defined in this DPAGE to be directed to both the left and right platens.

PD=

(for the 3180 and 3290 in partition formatted mode) Specifies the name of the partition descriptor of the partition associated with the DPAGE statement. This parameter maps a logical page of a message to or from the appropriate partition. The name of the PD must be contained within the PDB statement specified in the DEV statement.

ACTVPID=

(for the 3290 in partition formatted mode) Specifies the name of an output field in the message containing the partition identification number (PID) of the partition to be activated. This dfldname must be referenced by an MFLD statement and must not be used as the label of a DFLD statement in the DEV definition. The application program places the PID of the partition to be activated in this field. The PID must be in the format of a two byte binary number ranging from X'0000' to X'000F'.

Do not specify this operand for the 3180. Because only one partition is allowed for this device, you need not specify an active partition.

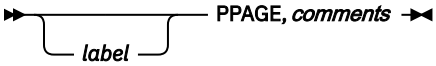
PPAGE statement

The PPAGE statement, valid only for device types of DPM-An or DPM-Bn, defines the beginning of a presentation page.

A presentation page is the unit of data delivered to the remote program in response to a paging request when OPTIONS=PPAGE has been specified in the DIV statement for this definition. For DPM-Bn MODE=RECORD only, if OPTIONS=MSG or DPAGE has been specified, paging is as described for those options under the DIV statement, and the PPAGE statement then defines the beginning of a new record (that is, it is equivalent to a RCD statement).

For an input DPAGE, only one PPAGE statement is allowed, and it must be placed between the DPAGE statement and the first DFLD statement. For an output DPAGE, if two consecutive PPAGE statements appear in the DPAGE for a message defined with OPTIONS=PPAGE, only an output message header with the PPAGE label as its data name is sent to the remote program, except OPTIONS=(PPAGE,DNM) for DPM-Bn. For DPM-Bn, a PPAGE statement without a DFLD statement is not allowed when OPTIONS=(PPAGE, NODNM) is specified for DIV TYPE=OUTPUT. A warning message is issued, and the PPAGE statement is ignored. For OPTIONS=MSG or DPAGE, consecutive PPAGE statements are ignored.

Format



Parameters

label

A one- to eight-character alphanumeric name should be specified. For OPTIONS=PPAGE, this label is sent as the data name for DPM-An or as the data structure name for DPM-Bn in the message output header or DD header to identify the data structure of this presentation page to the remote program. If no label is specified, MFS generates a diagnostic label that is sent to the remote program in the header.

Tip: Specify a user-defined label because the MFS-generated name can change whenever the MFS definitions are recompiled.

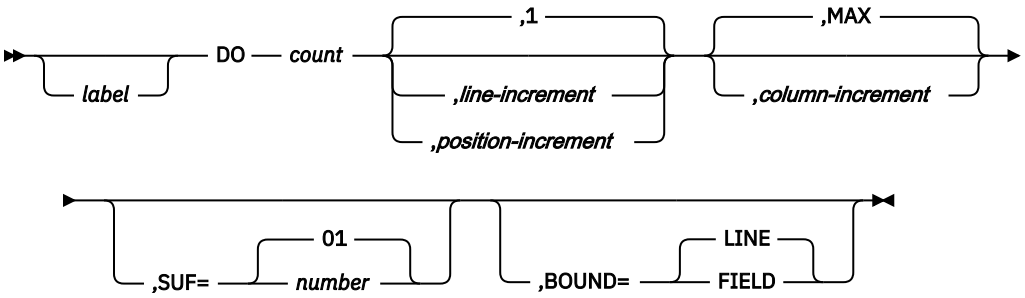
The label specified should be unique, at least within a given FMT definition, and preferably within an IMS system if the remote program uses this label to identify the appropriate DSECT for formatting the data included in this presentation page.

DO statement

The DO statement causes repetitive generation of DFLD and RCD statements between the DO and ENDDO statements.

When DO is used, there are restrictions in the naming of DFLDs.

Format



Parameters

label
A one- to eight-character alphanumeric name can be specified. It is not used.

count
Specifies how many times to generate the statements.

line-increment
Specifies how much to increase the line position after the first cycle. The first cycle uses the *lll* value specified in the POS= keyword of the DFLD statement. The default is 1. This parameter is not specified for DEV type DPM-An or DPM-Bn.

position-increment
Specifies how much to increase the position parameter after the first cycle. The first cycle uses the *nnn* value specified in the POS= operand of the DFLD statement. The position increment is used for an input device format when MODE=STREAM is specified. This parameter is not specified for DEV type DPM-An or DPM-Bn.

MAX
Specifies that the line increment to be used at the end of each cycle and the column values in the DFLDs are to remain the same for each cycle. This parameter is not used if MODE=STREAM is specified for the device format or if DEV type is DPM-An or DPM-Bn; if present, it is ignored.

column-increment
Specifies how much to increase the column position after the first cycle. The first cycle uses the *ccc* value specified in the POS= keyword of the DFLD statement. The default is MAX. This parameter is not used for DEV type DPM-An or DPM-Bn, or when MODE=STREAM is specified for the device format, because it is ignored.

SUF=
Specifies the 2-digit suffix to be appended to the *dfldname* of the first group of generated DFLD statements. The default is 01. MFS increments the suffix by one on each subsequent generation of statements.

If the specified suffix exceeds 2 digits, MFS uses the rightmost 2 digits.

If the specified count is such that the generated suffix eventually exceeds 2 digits, MFS reduces the count to the largest legitimate maximum value. For example, if count equals 8 and SUF=95, invalid suffixes of 100, 101, and 102 would result. In this instance, MFS reduces the count to 5, processes the statement, and issues an error message.

BOUND=

Specifies when updates to line position and column position are to occur. The default is LINE. This parameter is not used if MODE=STREAM is specified for the device format or if DEV type is DPM-An or DPM-Bn; if present, it is ignored.

LINE

Specifies that all fields be inspected before the repetition is performed. If the column increment would cause any field in the group of DFLD statements to not fit on a line, the column position value for all fields is reset to the initial value, and the line position values are increased by the line-increment value.

FIELD

Specifies that each time the statement is repeated, the column position value is increased by the column-increment value. If MAX is specified, or the new column position value reaches device line length capacity, the line position value is increased by the line-increment value and the column position value is reset to its initial value.

Example of line and column increment

The following example demonstrates how to increment lines and columns:

```
     DO      20,1,38
A1      DFLD    POS=(9,6),LTH=6
B1   DFLD    POS=(9,27),LTH=3
```

In this example, A1 and B1 are increasing by line increment (1) and column increment (38). Generation would proceed in the following fashion by a compiler:

- Add the column increment to each column value in the set, resulting in positions (9,44) and (9,65).
- Test to see if any field using these new column values would exceed the line size limitation for this device. In this example, assume a limitation of 80 for a 3270 Model 2.
- Since there is no violation of line width, generate A2 and B2 using the new column values and the same line value.
- Add the column increment again, resulting in positions (9,82) and (9,103).
- Since the fields would exceed line width, the column values are reset to the original values of (9,6) and (9,27) and the line increment is applied. The resulting positions are now (10,6) and (10,27).
- Generate A3 and B3 using the new line values, with column values as in the original statements.

Generation continues in this manner until the count of 20 iterations is reached.

Printing generated DFLD statements

The generated DFLD statements can be printed in a symbolic source format by specifying COMP in the parameter list of the EXEC statement. This provides a means of seeing the results of the DFLD statement generation without having to interpret the intermediate text blocks.

The following items are printed for each generated DFLD statement:

- The generated statement sequence number followed by a plus sign (+) to indicate that the DFLD statement was generated as a result of DO statement processing.
- The DFLD statement label, if present, including the appended suffix.
- The statement operator, DFLD.
- For EGCS literals, the G, SO, and SI are not present. Literals are truncated if there is insufficient room to print all specifications. Truncation is indicated by a portion of the literal with three periods (...), representing the truncated portion.
- ATTR=(YES,*nn*), if present.
- ATTR=YES, if present.

- ATTR=*nn*, if present.
- ATTR=(...), if attributes are present.
- EATTR=(...), if present.
- The RECORD or STREAM form of the POS= keyword, with the line and column or stream position updated by the respective increments. This is not printed if DEV type is DPM-An or DPM-Bn.
- SCA, if present.
- The field length, in the form of LTH=*nnnn*.

No other operands are printed, even if specified on the source DFLD statement.

For device type DPM-An or DPM-Bn, the RCD statement can appear between a DO and ENDDO statement. If it does, a new record boundary is created for each repetitive generation of the DFLD field following the RCD statement. For example, the following sequence causes the DFLDs A01, B01, and C01 to be in record 1, while A02, B02, and C02 are in record 2, and A03, B03, and C03 are in record 3.

```
     DO 3
     RCD
 A   DFLD    LTH=10
 B   DFLD    LTH=10
 C   DFLD    LTH=10
     ENDDO
```

Alternatively, the RCD statement can immediately precede the DO statement. If it does, a new record boundary begins with the first DFLD after the DO statement and does not end until the ENDDO statement (or the maximum record length) is reached. For example, the following sequence causes the DFLD D01 to begin a new record, in which E01, D02, and E02 also occur.

```
     RCD
     DO 2
 D   DFLD    LTH=10
 E   DFLD    LTH=10
     ENDDO
```

RCD statement

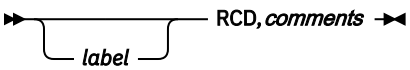
The RCD statement, valid for DEV TYPE=DPM-An or DPM-Bn only, can be used to influence the placement of DFLDs in records.

The RCD statement precedes a DFLD statement and initiates a new transmission record for delivery to a remote program. DFLDs following the RCD statement are included into the transmission record until the next RCD statement or the maximum record length is reached (or, if NOSPAN is specified, until a field will not be fully contained in the current record).

The RCD statement can be placed after the PPAGE, DO, DFLD, or ENDDO statements. If a RCD statement is immediately followed by another, only the first one is effective.

The RCD statement is invalid for STREAM mode.

Format



Parameters

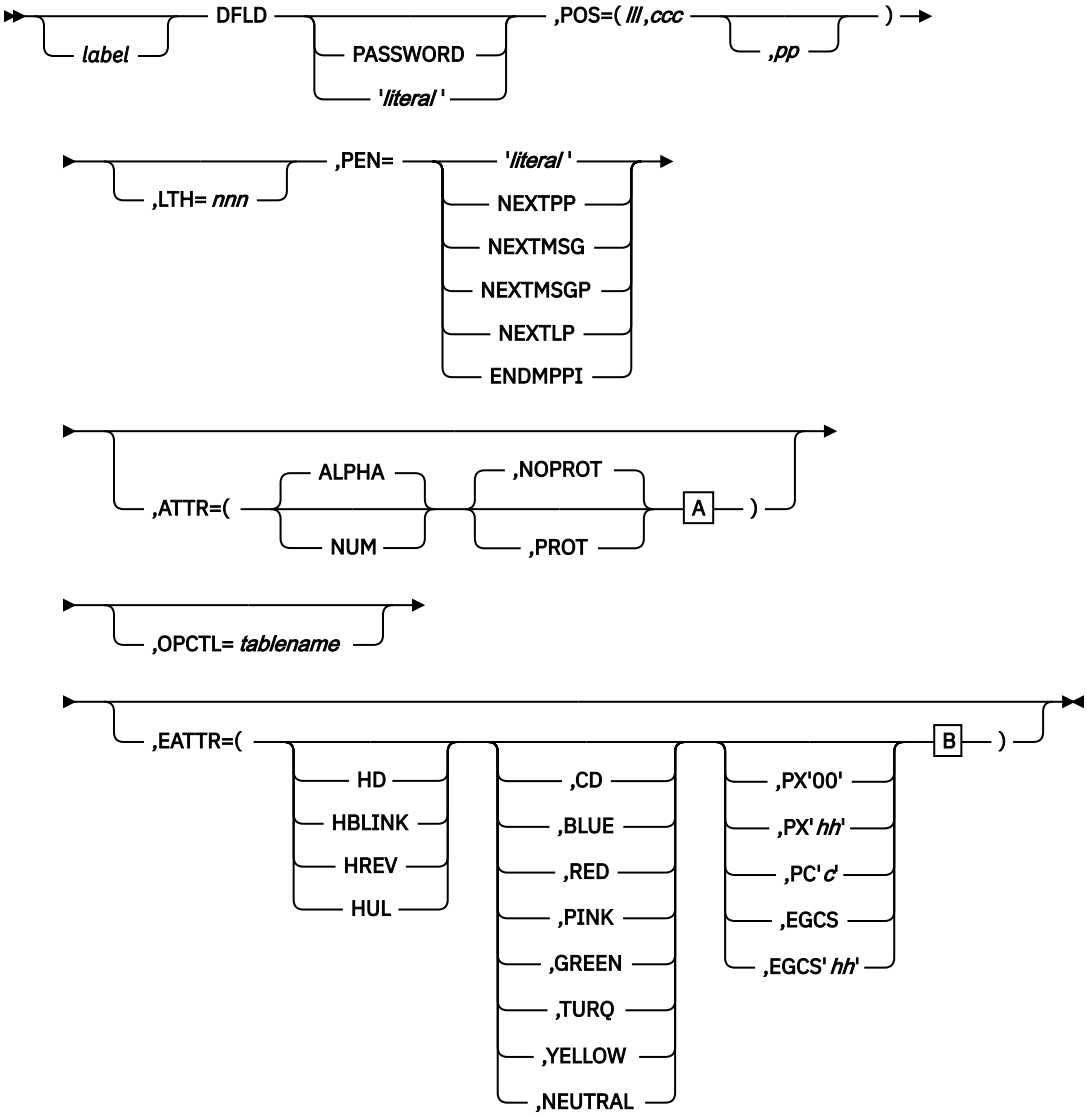
label
A one- to eight-character alphanumeric name can be specified. It is not used.

DFLD statement

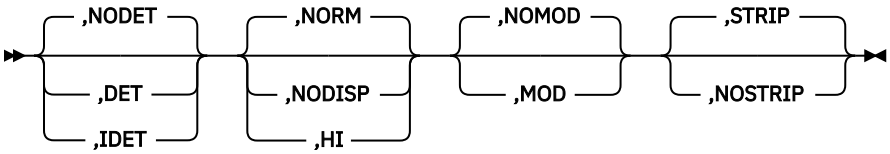
The DFLD statement defines a field within a device format which is read from or written to a terminal or remote program.

Only those areas which are of interest to the IMS or remote application program should be defined. Null space in the format does not need to be defined.

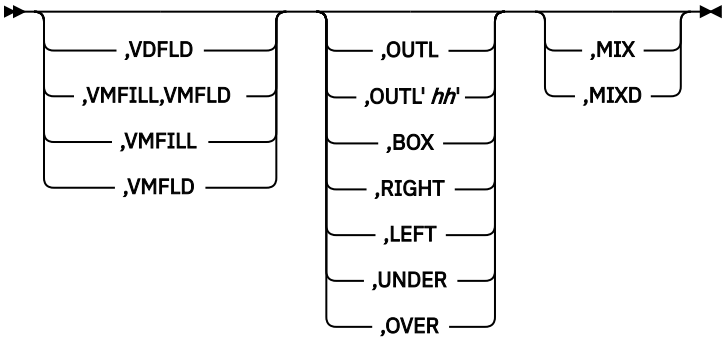
Format for DEV TYPE=3270 or 3270-An



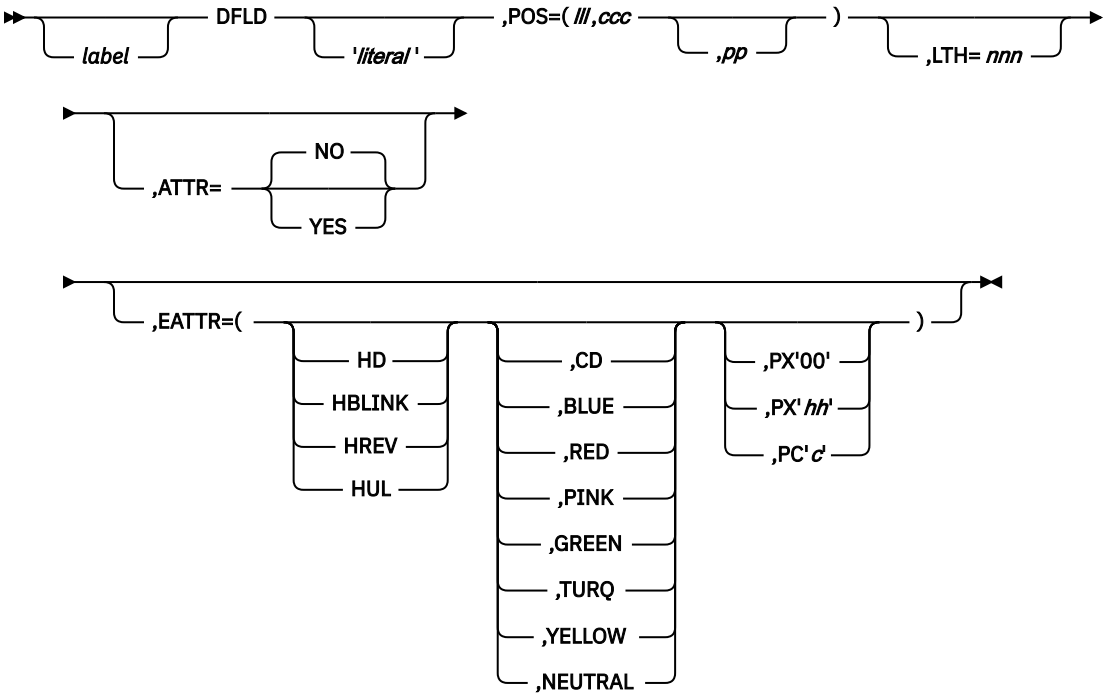
A



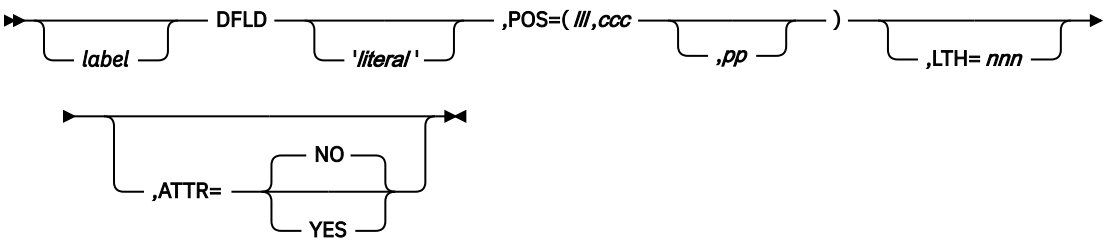
B



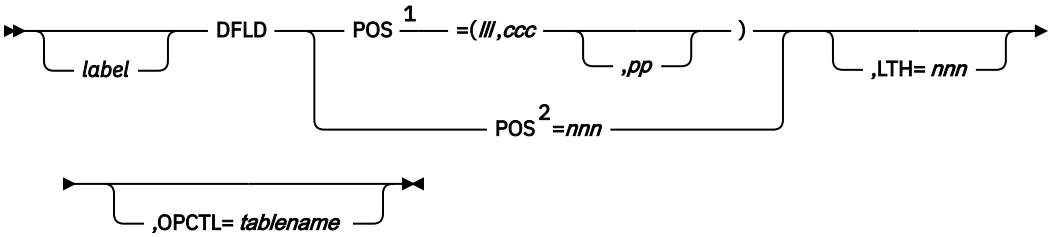
Format for DEV TYPE=3270P



Format for DEV TYPE=FIDS, FIDS3, FIDS4, FIDS7, FIFP, FIJP, and FIPB



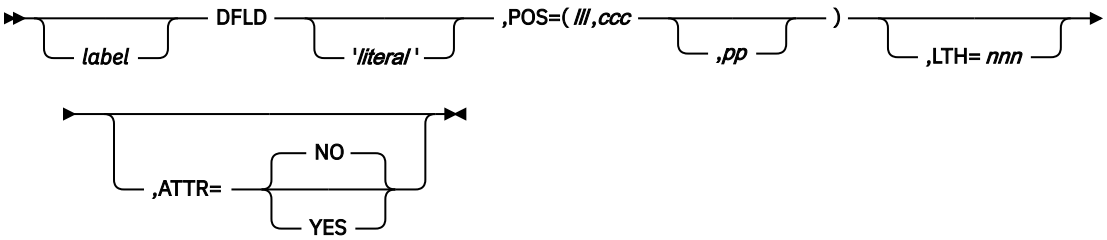
Format for DEV TYPE=FIN



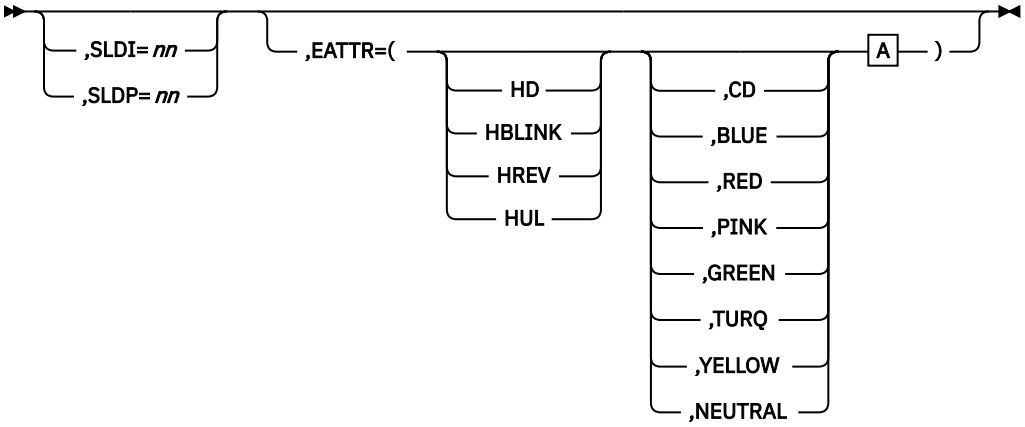
Notes:

[1] MODE=RECORD only
[2] MODE=STREAM only

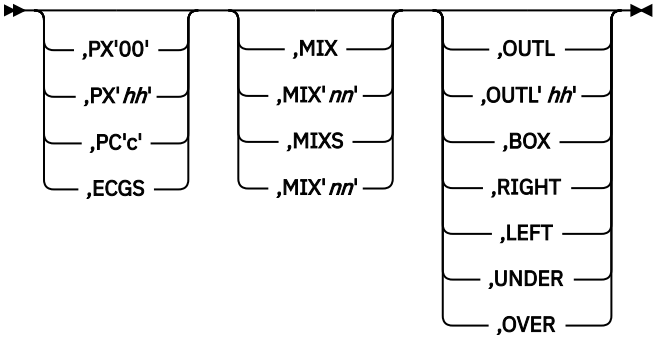
Format for DEV TYPE=SCS1 or SCS2 and DIV TYPE=OUTPUT



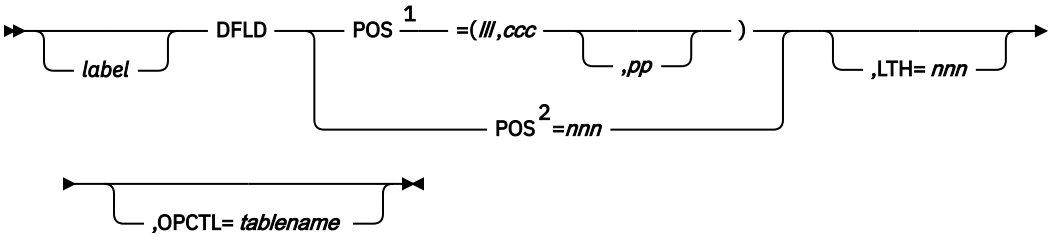
Format for SCS1 only



A



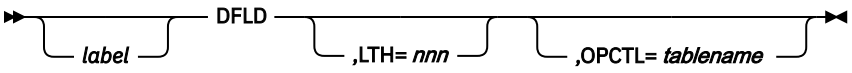
Format for DEV TYPE=SCS1 or SCS2 and DIV TYPE=INPUT



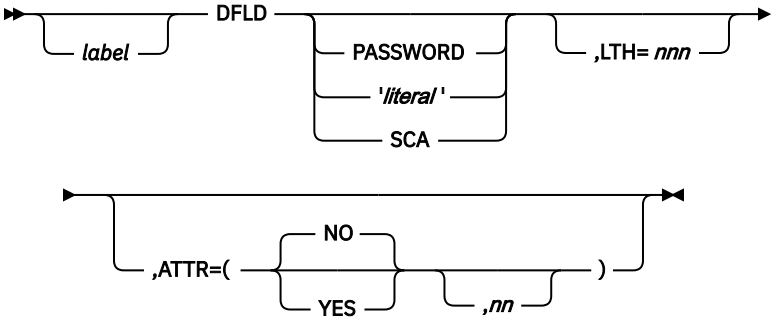
Notes:

[1] MODE=RECORD only
[2] MODE=STREAM only

Format for DEV TYPE=DPM-An or DPM-Bn and DIV TYPE=INPUT



Format for DEV TYPE=DPM-An or DPM-Bn and DIV TYPE=OUTPUT



Parameters

label
A one- to eight-character alphanumeric name can be specified. This label (dfldname) can be referred to by a message descriptor in transferring data to and from a terminal or remote program. If the repetitive generation function of MFS is used (DO and ENDDO statements), this dfldname should be restricted to 6 characters maximum length. When each repetition of the statement is generated, a 2-digit sequence number (01 to 99) is appended to the label. If the label specified here is greater than 6 characters and repetitive generation is used, the label is truncated at 6 characters, and a 2-digit sequence number is appended to form the 8-character name. No error message is provided if this occurs.

If PASSWORD, SCA, or '*literal*' is specified, label is not valid, and specification of a label will result in an error message. If a DPN, PRN, RDPN, or RPRN dfldname is specified on the DIV statement, the *dfldname* cannot be used as a DFLD label for the current DIV statement.

PASSWORD
Identifies this field as the location of the IMS password field for input messages.

IMS supports mixed case or upper case passwords up to 8 characters, and password phrases up to 100 characters.

Tip: Use the PASSWORD capability in the input message definition. If you specify PASSWORD you cannot refer to the field described by this DFLD statement with a message descriptor. Additionally, if you specify PASSWORD you must omit *label*.

'literal'
Specifies a literal character string to be presented to the device. The length of *literal* cannot exceed 256 bytes for 3270 display devices, 40 bytes for FIDS and FIDS3, 64 bytes for FIDS4, 80 bytes for FID57, 256 bytes for 3270P, and line width for all printer and punch devices. For DPM, the length of *literal* cannot exceed the value specified in the RCDCTL operand.

For 3270 displays, literal fields have the PROT attribute whether specified or not; the NUM attribute is assumed if ALPHA is not specified.

Restriction: If you specify *literal* you cannot refer to the field described by this DFLD statement with a message descriptor. Additionally, if you specify *literal* you must omit *label*.

SCA
Specifies, for DPM definitions only, that SCA information, when sent by the IMS application program or specified in the DSCA, is to be sent in this DFLD.

If SCA is specified, *label* must not be specified.

POS=
Defines the first data position of this field in terms of line (lll), column (ccc), and physical page (pp) of the display format. If pp is omitted, 1 is assumed.

For DEV TYPE=FIN,FIDS,FIDS3,FIDS4, FIDS7,FIJP,FIPB,FIFP,SCS1, or SCS2

lll,ccc
Specifies the record number and position within the record of this field. This form is required if MODE=RECORD. lll and ccc must be greater than or equal to 1.

nnn
Specifies the starting position of this field in STREAM mode input. If not specified, this field starts immediately following the preceding field, or at the left margin if this is the first field. If MODE=STREAM has been specified, and POS= is specified, this form is required. *nnn* must be greater than or equal to 1.

lll,ccc,pp
Specifies the line, column, and optionally, the physical page number for an output field. lll, ccc, and pp must be greater than or equal to 1.

For DEV TYPE=3270, 3270-An, or 3270P

lll,ccc,pp
Specifies the line, column, and optionally, the physical page number for an output field. *lll*, *ccc*, and *pp* must be greater than or equal to 1.

For 3270 displays, POS=(1,1) must not be specified. Fields must not be defined such that they wrap from the bottom to the top.

Restriction: On some models of 3270s, the display screen cannot be copied when a field starting on line 1, column 2, has both alphabetic and protect attributes.

For DEV TYPE=DPM-An or DPM-Bn

For DPM devices
The POS= keyword is ignored.

LTH=
Specifies the length of the field. This operand should be omitted if *'literal'* is specified in the positional parameter, in which case the length of *literal* is used as the field length. Unpredictable output formatting can occur if this operand is used in conjunction with a *'literal'* and the two lengths are different. The specified LTH= cannot exceed the physical page size of the device.

The maximum allowable length for all devices except 3270, 3604 display, and DPM with RCDCT=NOSPAN is 8000 characters. For 3270 displays, the maximum length is one less than screen size. For example, for a 480-character display, the maximum length is 479 characters. For a FIDS display component, the maximum length is 240 characters; for a FIDS3, the maximum length is 480 characters; for a FIDS4, the maximum length is 1024 characters; for a FIDS7, the maximum length is

1920. A length of 0 must not be specified. For DPM, if RCDCT=NOSPAN is specified, the length must be less than or equal to the RCDCTL value, if RCDCTL is less than 8000. If SCA and LTH= are both specified, LTH must be 2.

POS= and LTH= do not include the attribute character position reserved for a 3270 display device or a DFLD with ATTR=YES specified. The inclusion of this byte in the design of display/printer formats is necessary because it occupies the screen/printed page position preceding each displayed/printed field even though it is not accessible by an application program.

When defining DFLDs for 3270 printers, a hardware ATTRIBUTE character is not used. Therefore, fields must be defined with a juxtaposition that does not allow for the attribute character unless ATTR=YES is specified. However, for printers defined as 3270P the last column of a print line (based on FEAT=, WIDTH=, or the device default width) cannot be used. The last column of the line is reserved for carriage control operations performed by IMS. Thus, if the print line specifies 120 (FEAT=120) and the DFLD specifies POS=(1,1),LTH=120 then 119 characters are printed on line 1 and one character on line 2.

For DPM definitions, if OPTIONS=NOSIM2 is specified on the DIV statement, and ATTR=YES or YES,*nn* is specified, 2 bytes plus the extended attributes are added to the length of the DFLD. The first two bytes are reserved for the binary 3270 attribute, (protect, numeric, and so forth.) If OPTIONS=SIM is specified, 1 byte or 1 byte plus the extended attributes is added to the length of the DFLD with ATTR=YES or YES,nn. The first byte of the field is thus reserved for the simulated attribute.

Detectable fields (DET or IDET) must include four positions in POS and LTH for a 1-byte detection designator character and 3 pad characters, unless the detectable field is the last field on a display line, in which case only one position for the detection designator character is required. The detection designator character must precede field data, and pad characters (if required) follow field data. Detection designator and required pad characters must be supplied by the application program or MFLD literal with the field data. Pad characters can also be required in the preceding field on the device.

ATTR=

Defines the display attributes of this field for each of the listed DEV TYPE, DIV TYPE combinations:

- For DEV TYPE=3270 or 3270-An

  Attribute keywords can be specified in any order and only those desired need be specified. The underlined keywords do not have to be specified, because they are defaults.

  When two user-defined fields are separated by two or more characters, MFS generates an undefined field to represent that space in the display buffer. The display attributes for an undefined field are NUM, PROT, and NODISP.

  ALPHA | NUM

  Specifies whether the field should have the numeric attribute. The numeric attribute specifies that the Numeric Lock feature (automatic upshift of data entry keyboard) will be used by the 3275/3277 or 3276/3278. If NUM and PROT are specified for the field, the auto-skip feature is used. That is, upon entry of a character into the last character location of an unprotected field, the cursor automatically skips the field with the NUM and PROT attribute specifications and is positioned to the first character location of the next unprotected field. If an undefined field, as described in the ATTR= parameter, follows the filled unprotected field, the auto-skip feature is used. This parameter, in conjunction with the PROT parameter, is used to lock the COPY function. See "PROT" for details.

  NOPROT | PROT

  Specifies whether the field is protected from modification by you. For literal fields, PROT is used and specification of NOPROT is ignored.

  The IMS copy function on remote 3270 terminals can be locked by setting the attribute value of protect and alpha for an attribute byte in line 1 and column 1 of a display. When the copy function is locked, it cannot be used to copy the contents of a display to a printer. The "Local Copy Function" available on the 3274 and 3276 control units is not locked by the attribute setting. The "Local Copy Function" is invoked by the print key.

NODET|DET|IDET
Specifies the detectability of the field through light pen operations. DET specifies a deferred detectable field, while IDET indicates an immediately detectable field. You must provide appropriate designator and pad characters as discussed under the LTH= operand. Note that the 3270 display devices place restrictions on the number of detectable or mixed detectable and nondetectable fields that can precede that last detectable field on a given line.

NORM|NODISP|HI
Specifies the field's display intensity as normal (NORM), high intensity (HI), or nondisplayable (NODISP). If NODISP is specified, DET or IDET cannot be specified.

When defining a high-intensity (HI) field, including a detection designator character as the first data byte causes the high-intensity (HI) field to be detectable.

NOMOD|MOD
defines whether or not the field-modified-attribute byte should be assumed for this field. MOD causes the terminal to assume the field has been modified by you even though it was not (that is, the modified data tag (MDT) is set in the field-modified-attribute byte). This should not be confused with the PROT attribute which prevents modification by you. MOD is ignored for literal fields.

When MOD is specified, each time MFS sends output for this physical page, the modified attribute is set (unless overridden by dynamic attribute modification).

STRIP|NOSTRIP
Specifies whether the pen detect designator byte preceding the input field should be stripped (STRIP) before presentation to the application program. If an EGCS attribute is defined for a light-pen-detectable field, you should specify ATTR=NOSTRIP on the DFLD statement and design the application program to bypass or remove the two designator characters from the input data. If ATTR=STRIP is specified or defaulted, MFS will only remove the first designator character and the last character in the field could be lost (truncated).

- For DIV TYPE=OUTPUT and DEV TYPE=3270P, FIDS, FIDS3, FIDS4, FIDS7, FIFP, FIJP, FIPB, FIS1, or SCS2

  Attribute keywords specify whether (YES) or not (NO) the first byte of this field will be used to display attribute information when the output message includes attribute information for the field. The default is NO. If ATTR=YES is specified, the LTH= and POS= keywords do not have to allow for the simulated attribute byte because the MFS preprocessor adjusts the keyword values internally. The action taken when ATTR=YES is specified is:

  CURSOR
  (FIDS, FIDS3, FIDS4, and FIDS7 ABSOLUTE output only). The cursor will be positioned to the first position of this field.

  NODISP
  No data sent regardless of other attributes

  HI
  An asterisk (*) is placed in the first byte

  MODIFIED
  An underscore character (_) is placed in the first byte

  HI and MODIFIED
  An exclamation point (!) is placed in the first byte

  If attribute information is not provided from the output message, the first byte is a blank.

- For DIV TYPE=OUTPUT, DEV TYPE=DPM-An, and DEV TYPE=DPM-Bn, 3270P, FIDS, FIDS3, FIDS4, FIDS7, FIFP, FIJP, FIPB, FIS1, or SCS2

  Attribute keywords specify whether (YES) or not (NO) the first one or two bytes of this field carries existing 3270 attributes and whether extended attributes (nn) are present. The keywords can be used in various combinations as follows:

YES
Specifies that the first one or two bytes of this field are used to convey the existing 3270 attributes (in simulated or binary form depending upon the specification of SIM or NOSIM2 respectively on the DIV statement) from the IMS application program to the remote program. (SIM causes MFS to simulate an attribute. NOSIM2 causes MFS to pass the bits exactly as entered.)

Thus, if ATTR=YES is specified and OPTIONS=SIM or OPTIONS= is not specified, one byte is added to the length of the DFLD. If OPTIONS=NOSIM2, two bytes are added to the length of the DFLD. These bytes are reserved as the attribute bytes to be transmitted to the remote program.

NO
Specifies that the first one or two bytes of this field will not be used to convey the existing 3270 attributes (in simulated or binary form respectively) from the IMS application program to the remote program. This is the default.

nn
Is the number of extended attributes that can be dynamically modified, and is a number from 1 to 4. An invalid specification is defaulted to 1. Two additional bytes are added to the length of the DFLD for each attribute specified (2 x *nn*). The additional bytes, which just precede the data, either can (YES) or must not (NO) follow the bytes reserved for the existing 3270 attribute bytes. These bytes are used to convey the extended attributes (in binary form) from the IMS application program to the remote program. The attributes are always transmitted as presented from the IMS application program. They are never simulated or validated.

YES,*nn*
When used in combination, YES,*nn* specifies that both attributes and extended attributes are to be transmitted. In this case, and depending upon the specification of SIM and NOSIM2 as described:.

When specified with SIM, specifies that 3270 simulated attributes (1 byte) plus extended attributes (2 x *nn* bytes) of this field are to be transmitted from the IMS application program to the remote program. The total number of bytes used to convey all of these attributes to the remote program is 1 + (2 x *nn*)

When specified with NOSIM2, specifies that 3270 attributes in binary form (2 bytes) plus extended attributes (2 x *nn* bytes) of this field are to be transmitted from the IMS application program to the remote program. The total number of bytes used to convey all of these attributes, which are all in binary form, to the remote program is 2 + (2 x *nn*).

NO,*nn*
When used in combination, NO,*nn* specifies that only extended attributes are transmitted. Thus, the number of bytes transmitted, in binary form, is (2 x *nn*) only.

Valid specifications and the number of bytes that must be reserved are:

```
For  DIV ,OPTION=NOSIM2  then:
DFLD    ,ATTR=(YES,nn)     2 + (2 × nn)
DFLD    ,ATTR=(NO,nn)      2 × nn
DFLD    ,ATTR=(,nn)        2 × nn
DFLD    ,ATTR=YES          2
DFLD    ,ATTR=NO           0
For  DIV ,OPTION=SIM or not specified  then:
DFLD    ,ATTR=(YES,nn)     1 + (2 × nn)
DFLD    ,ATTR=(NO,nn)      2 × nn
DFLD    ,ATTR=YES          1
DFLD    ,ATTR=NO           0
```

EATTR=
Is valid for output DFLDs only and defines the extended attributes of this field for DEV TYPE=3270, 3270-An, 3270P, or SCS1.

Not all extended attributes apply to all device types. To ensure that your specifications for your device types are correct, refer to the component description manual for your device.

The operands specify:

- Additional field highlighting
- Field color
- Field outlining
- Input control
- Validation to be performed
- Local ID of the programmed symbol buffer

Characters are selected from the programmed symbol buffer and placed in the field. These operands can be specified in any order. When the device default value is selected for an operand, it is used to hold a place in the data stream to permit application program modification of the attribute so specified.

To specify the additional highlighting for the field use the following:

HD
device default

HBLINK
blink

HREV
reverse video

HUL
underline

To specify the field's color use the following:

- **BLUE**
- **RED**
- **PINK**
- **GREEN**
- **TURQ(uoise)**
- **YELLOW**
- **CD**
- **NEUTRAL**

The last two operands are used as follows:

CD
Used to specify the default.

NEUTRAL
Used to specify device-dependent. The particular color displayed for NEUTRAL is device-dependent. In general, NEUTRAL is white on displays and black on printers with single-plane programmed symbols and as multicolored on displays or printers with tri-plane programmed symbols.

The following five operands—PX'00', PX'*hh*', PC'*c*', EGCS, and EGCS'*hh*'—are mutually exclusive. That is, a field can be specified as having one of these characteristics, but not a combination thereof. For all 3270 devices, MFS does not verify that any specified character set has been properly loaded. The programmed symbol buffers can be loaded by an IMS application program using the MFS bypass.

PX'00'|PX'*hh*'|PC'*c*'
Specifies a value that must correspond to the local ID specified for a programmed symbol buffer already loaded or to the EGCS programmed symbol buffer.

PX'00'
Is the same as no specification, except that it allows an application program to specify a programmed symbol buffer for the field through dynamic modification of the programmed symbol attribute.

PX'*hh*'
Is a hexadecimal character in the range X'40' through X'FE'.

PC'*c*'
Is a hexadecimal character within the range X'40' through X'FE'.

EGCS|EGCS'*hh*'
Is valid only on output DFLDs for the 3270 display. SCS1 device types can specify EGCS only and not EGCS '*hh*'.

When an extended graphic character set literal is specified on a DFLD statement, the extended graphic character set attribute is forced—that is, you do not have to code EATTR=EGCS'*hh*' for 3270 displays or EATTR=EGCS for SCS1 device types. For 3270 displays, a programmed symbol value of X'F8' is set.

Restriction: The IMS application program cannot modify the SCS1 DFLD extended graphic character set attribute.

When defining an EGCS field for a 3283 Model 52, the length must be an even number. If the EGCS field spans device lines, WIDTH= and POS= should be specified so that an even number of print positions are reserved on each of the device lines.

EGCS
Specifies the field attribute for the field as Extended Graphic Character Set. Also specifies the field attribute for the field as Double Byte Character Set.

EGCS'*hh*'
'*hh*' is the programmed symbol value that is used. The value for '*hh*' can be any hexadecimal value from X'40' through X'FE' or X'00'. If '*hh*' is omitted from the extended graphic character set specification for a 3270 display, a programmed symbol value of X'F8' is assumed. '*hh*' is ignored if specified for an SCS1 device.

To define an EBCDIC field that can be dynamically modified by the IMS application program to accept extended graphic character set data, the programmed symbol attribute should be specified as EGCS'00'.

VDFLD|VMFILL|VMFLD|VMFILL,VMFLD
Defines the type of validation for the field as follows:

VDFLD
Default

VMFILL
Mandatory fill

VMFLD
Mandatory field

VMFILL,VMFLD
A combination of mandatory fill and mandatory field

If a field is defined as protected (ATTR=PROT) or if it is a literal with validation attributes specified, then the validation attribute specifications are reset and a message is issued.

The following are used to specify field outlining:

OUTL'*hh*'
Field outlining with field outlining value '*hh*'

OUTL
Device default

BOX
Box

RIGHT, LEFT, UNDER, OVER
Lines that can be specified individually or in combination

Field outlining value '*hh*' is a two-digit hexadecimal number between X'00' and X'0F'. If any other value is specified, the device default, X'00', is assumed. The following table shows the values for the field outlining patterns.

Table 20. Field outlining values

Value	UNDER	RIGHT	OVER	LEFT
00				
01	X			
02		X		
03	X	X		
04			X	
05	X		X	
06		X	X	
07	X	X	X	
08				X
09	X			X
0A		X		X
0B	X	X		X
0C			X	X
0D	X		X	X
0E		X	X	X
0F	X	X	X	X

Field outlining for 3270 displays and SCS1 printers can be dynamically modified by code in an application program. The position of left, right, over, and underlines differ according to the device.

The following is a brief description of field outlining for the IBM 5550 family (as 3270) of devices.

3270 display
Left and right lines are printed in the position of the 3270 basic attribute byte. The overline of the current line and the underline of the preceding line are the same line.

The underline for the 24th line is the same line as the line separating the application program area and your message area.

SCS1 printer
Left and right lines are printed in the byte reserved by MFS before and after the current field. The overline of the current line and the underline of the preceding line are the same line. When an underline is specified in the last line of the page, an underline is drawn in the last line of the page, and an overline is drawn on the first line of the next page.

If one byte space exists between two adjacent fields, the right line of the first field is the same line as the left line of the second field.

MIX|MIXD|MIX'*nn*'|MIXS|MIXS'*nn*'
Specify a DBCS/EBCDIC mixed field.

3270 display

MIX
DBCS/EBCDIC mixed field

MIXD
device default

Input control for the 3270 display can be dynamically modified by the application program.

SCS1 printer

MIX
DBCS/EBCDIC mixed field with SO/SI blank print option.

MIXS
DBCS/EBCDIC mixed field with SO/SI blank print suppress option.

MIX'*nn*'
'*nn*' is the maximum number of SO/SI pairs. DBCS/EBCDIC mixed field with SO/SI blank print option.

MIXS'*nn*'
'*nn*' is the maximum number of SO/SI pairs. DBCS/EBCDIC mixed field with SO/SI blank print suppress option.

The '*nn*' is buffer information used by MFS message editor and must be a two-digit decimal number between 01 and 31. If MIX or MIXS is specified, the MFS default is calculated as follows:

MIX
DFLD length divided by 5 plus 1, or 31, whichever is smaller.

MIXS
DFLD length divided by 3 plus 1, or 31, whichever is smaller.

When a field spans continuation lines, the number '*nn*' obtained from the field length with either of the methods plus 1, is assigned to each line.

With the SCS1 printer, when DBCS/EBCDIC mixed data spanning across continuation lines is split at a DBCS character, MFS replaces the last character with a blank and places that character at the beginning of the next line. As a result, one print position is lost.

PEN=
Specifies a literal to be selected or an operator control function to be performed when this field is detected. If (1) *'literal'* is specified, (2) the field is defined as immediately detectable (ATTR= operand), and (3) contains the null or space designator character, the specified literal is placed in the field referred to by the PEN operand of the preceding DEV statement when the field is detected (if no other device fields are modified). If another field on the device is modified, a question mark (?) is provided instead of the literal. Literal length must not exceed 256 bytes.

If (1) a control function is specified, (2) the field is defined as immediately detectable (ATTR= operand), and (3) contains the null or space designator character, the specified control function is performed when the field is detected and no other device fields are modified. If another field on the device is modified, a question mark (?) is provided and the function is not performed. Control functions that can be specified are:

NEXTPP—PAGE ADVANCE
Specifies a request for the next physical page in the current output message. If no output message is in progress, no explicit response is made.

NEXTMSG—MESSAGE ADVANCE
specifies a request to dequeue the output message in progress (if any) and to send the next output message in the queue (if any).

NEXTMSGP—MESSAGE ADVANCE PROTECT
Specifies a request to dequeue the output message in progress (if any), and send the next output message or return an information message indicating that no next message exists.

NEXTLP—NEXT LOGICAL PAGE
Specifies a request for the next logical page of the current message.

ENDMPPI—END MULTIPLE PAGE INPUT
Specifies the end of a multiple physical page input message.

ENDMPPI is valid only if data has been received and will not terminate multiple page input (MPPI) in the absence of data entry.

OPCTL=
Specifies the name of a table, defined by a TABLE statement, that is to be checked for operator control requests when this device field is received. OPCTL processing occurs when the input device data is processed. If a control function is selected, in most cases the control function is performed immediately; no IMS input message is created.

SLDI=
For SCS1 printers, specifies the line density for an output message in lines per inch. (See also SLDP=.) SLDI= can also be specified on the DEV statement. SLDI= is validated for a value from 1 through 72. The value specified must be consistent with the architecture of the device for which this value is specified (see the appropriate device or component manual).

If SLDI= is specified both on the DEV statement and the DFLD statement, two SLD data streams are created. One is sent at the beginning of a message to set the line density. The second is sent within the message, just prior to the field on which the SLDI= specification is encountered, but after any vertical tabs and new line characters. The SLDI= specification within the message changes the line density from that set at the beginning of the message, and this latter line density remains in effect until explicitly reset.

SLDP=
For SCS1 printers, specifies the line density for an output message in points per inch. (See also SLDI=.) SLDP= can also be specified on the DEV statement. SLDP= is validated for a value from 1 through 72. The value specified must be consistent with the architecture of the device for which this value is specified (see the appropriate device or component manual).

If SLDP= is specified both on the DEV statement and the DFLD statement, two SLD data streams are created. One is sent at the beginning of a message to set the line density. The second is sent within the message, just prior to the field on which the SLDP= specification is encountered, but after any vertical tabs and new line characters. The SLDP= specification within the message changes the line density from that set at the beginning of the message, and this latter line density remains in effect until explicitly reset.

Attention: Be careful, when defining set line density (SLDx) keywords, to ensure that forms alignment is maintained. If SLDx= is improperly defined, the forms might not align properly. Also, note that SLDI= and SLDP= are mutually exclusive. Neither SLDI= nor SLDP= can occur on a DFLD statement between a DO and an ENDDO statement.

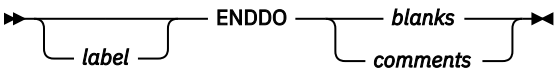
Related reference
MFS output message formats (Application Programming)

ENDDO statement

The ENDDO statement terminates the group of DFLD statements that are to be repetitively generated.

The generated DFLD statements are printed immediately following the ENDDO statement. An ENDDO statement is required for each DO statement entered in this definition.

Format



Parameters

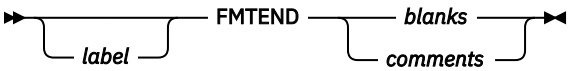
label
A one- to eight-character alphanumeric name can be specified. It is not used.

FMTEND statement

The FMTEND statement terminates a device format definition and is required as the last statement in the device format definition.

If this is the end of the input to SYSIN processing, the FMTEND statement must be followed by an END compilation statement.

Format



Parameters

label
A one- to eight-character alphanumeric name can be specified. It is not used.

Partition set definition statements

Partition set definition statements include the PDB statement, the PD statement, and the PDBEND statement.

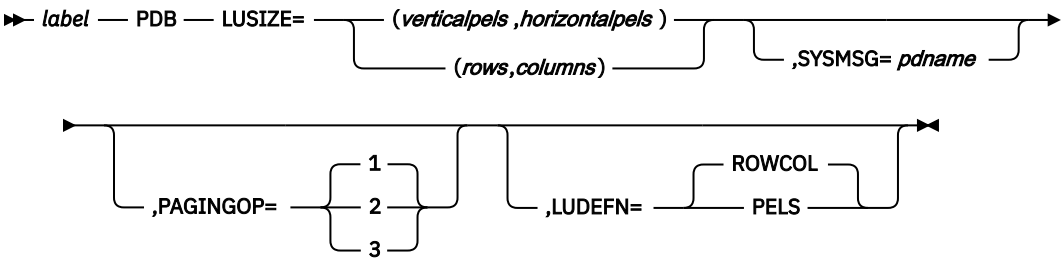
PDB statement

The PDB statement initiates and defines a partition set (a Partition Descriptor Block) for 3290 and 3180 devices in partitioned format mode.

The PDB statement contains several parameters that describe certain characteristics of the entire partition set. Its name is referenced by the PDB keyword of a DEV statement if a partition set is required to format logical pages of a message.

At least one PD statement must be specified within each PDB. Note, however, that for a 3180 in partitioned format mode, only one PD statement should be specified within each PDB. This is because only one partition can be specified for the 3180. There are additional differences in specifications that can be made for the partitioned 3180 and 3290.

Format



Parameters

label
A one- to eight-character alphanumeric name (pdbname) for the PDB must be specified.

LUSIZE=
Describes the physical size of the Logical Unit display for which the PDB is defined. If LUDEFN=PELS, the size is specified in picture elements (pels). If LUDEFN=ROWCOL, the size is specified in rows and columns (this is the default value). For the 3180, LUSIZE must be specified in terms of rows and columns.

SYSMSG=
Specifies the partition name (*pdname*) for displaying system messages. The system message partition should have only one field defined. This DFLD should be defined as at least LTH=79 so the system message is not truncated.

If the current PDB defines a system message partition, then all system messages are directed to this partition. If a system message partition is not defined, but a SYSMSG field is defined in the current DOF, the system message is directed to the system message field of the active partition. Finally, if the current PDB does not define a partition for system messages and the DOF does not define a field for that purpose, a system message destroys the current partitioned format mode and the 3290 returns to standard format mode.

PAGINGOP=
Specifies the option number (1, 2, or 3) for the partition page presentation algorithm. These three algorithms specify different ways of presenting the initial pages of the message to the partitions of the partition set. They also specify what paging actions result when you enter paging requests from the 3290 device.

The default of 1 must be accepted or specified on this operand for 3180 formats.

LUDEFN=
Indicates whether the LUSIZE parameter in the PDB statement and the VIEWLOC parameter in the PD statements are specified in rows and columns or in pels. LUDEFN is optional if all the PD statements use the same cell size and the default (ROWCOL) is acceptable. Note that ROWCOL must be specified or accepted as the default for 3180 formats.

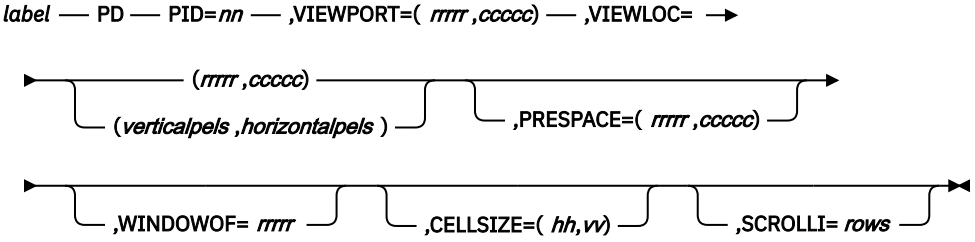
If two or more PD statements within the same PDB specify different cell sizes, PELS must be chosen.

PD statement

The PD statement defines one partition and its presentation space.

Every partition set described by a PDB statement must contain at least one PD statement. Note, however, that for a 3180 in partitioned format mode, only one PD statement should be specified within each PDB.

Format



Parameters

label
A one- to eight-character alphanumeric name (pdname) must be specified. This name is referenced by the DPAGE statement to associate a logical page with its appropriate partition.

PID=
Specifies a partition identifier number for the partition. Values 00 through 15 are valid for 3290 formats. Each partition must have a unique PID. A value of 00 must be specified for 3180 formats, because only one partition need be identified.

VIEWPORT=
Specifies the size of the viewport for the partition. *rrrrr* indicates rows and *ccccc* indicates columns. For the 3180 device, the following restrictions apply:

- If the number of columns is greater than or equal to 80, then the number of rows must be less than or equal to 43.
- If the number of columns is greater than 80 and less than or equal to 132, then the number of rows must be less than or equal to 27.

VIEWLOC=
Specifies the location of the viewport on the display screen, in terms of the distance offset from the top left of the screen. When the LUDEFN parameter of the PDB statement is ROWCOL, the distance is expressed in rows and columns. *rrrrr* indicates rows and *ccccc* indicates columns. When the LUDEFN parameter is PELS, the distance is expressed in the number of pels from the top of the screen and the number of pels from the left of the screen. When defining formats for the 3180, VIEWLOC must be expressed in rows and columns.

PRESPACE=
Indicates the size of the presentation space buffer in rows and columns. *rrrrr* indicates rows and *ccccc* indicates columns. If this parameter is not specified, the default is the size of the viewport specified on the VIEWPORT parameter. When this parameter is specified, the columns parameter is optional and defaults to the columns specification on the VIEWPORT parameter. If columns are specified, they must be the same as the columns specified in the VIEWPORT parameter.

When specifying this operand for 3180 formats, the product of the number of rows times the number of columns might not be greater than 7680.

WINDOWOF=
Indicates the initial offset in rows of the top edge of the view window from the top of the presentation space. The window maps the portion of the presentation space to be displayed onto the viewport on the screen. During interactive processing, change the offset by scrolling. The default value of WINDOWOF is zero.

CELLSIZE=
Indicates the number of horizontal and vertical pels in a character cell. Note that this specification is in an unusual order for MFS. That is, the width of the character cell is specified first, then the height. This is the reverse of the usual MFS order.

For the 3290, the default is 6 X 12 PEL (for a small character). Valid values for the 3290 are 6 X 12 to 12 X 31, or the value 00 X 00. If the value is 00 X 00, the 3290 device will select a cell size for optimum readability. This prevents MFS from making validity checks on the viewport locations and possible overlaps. Therefore, be careful to choose viewport size and location specifications accurately.

For the 3180, this operand should be specified according to usable screen area size as follows:

- CELLSIZE=(12,12)
  - 24 x 80
  - 32 x 80
  - 43 x 80
- CELLSIZE=(10,16)
  - 27 x 132

SCROLLI=
Indicates the number of rows that are scrolled when the scrolling function is used. The default scrolling increment is one row. If the scrolling increment is larger than the viewport size, part of the presentation space is not viewable on the screen. Specifying 0 as the scrolling increment disables the scrolling function.

PDBEND statement

The PDBEND statement terminates a partition set definition (a partition descriptor block) and is required as the last statement of the definition.

If this is the end of the input to SYSIN processing, the PDBEND statement must be followed by an END compilation statement.

Format

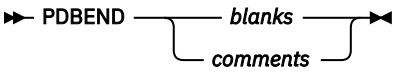


Table definition statements

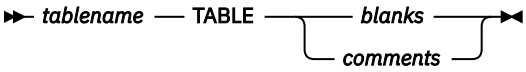
Table definition statements include the TABLE statement, the IF statement, and the TABLEEND statement.

TABLE statement

The TABLE statement initiates and names an operator control table that can be referred to by the OPCTL keyword of the DFLD statement.

The TABLE statement, and the IF and TABLEEND statements that follow, must be outside of a MSG or FMT definition.

Format



Parameters

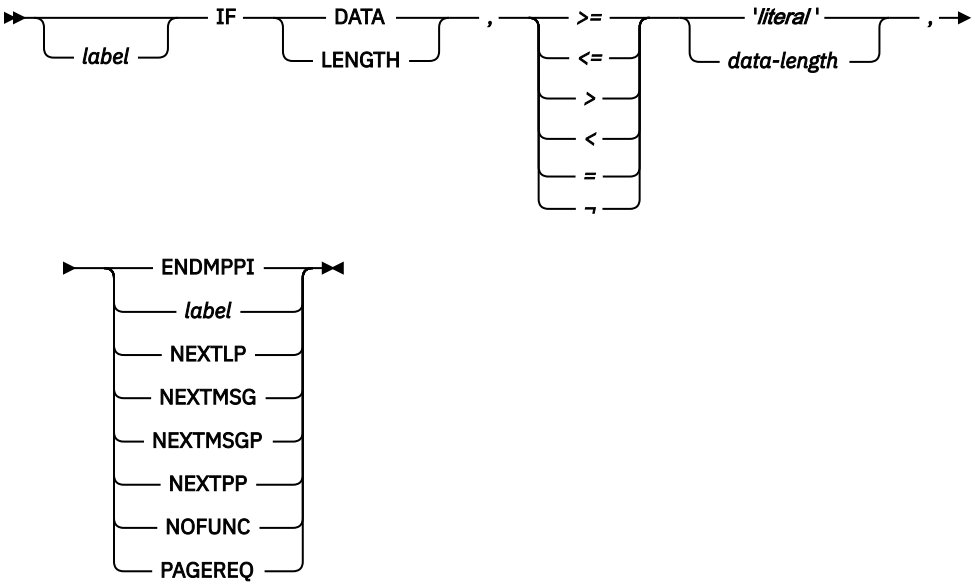
tablename
A 1- to 8-byte alphanumeric name for the table must be specified.

IF statement

The IF statement defines an entry in the table named by the previous TABLE statement.

Each IF statement defines a conditional operation and an associated control or branching function to be performed if the condition is true.

Format



Parameters

label
A one- to eight-character alphanumeric name can be specified. This label is required if a previous IF statement contained a branch function.

DATA
Specifies that the conditional operation is to be performed against the data received from the device for the field.

LENGTH
Specifies that the conditional operation is testing the number of characters entered for the field. The size limit for this field is the same as for DFLDs.

=, <, >, ¬, ≤, ≥
Specify the conditional relationship that must be true to invoke the specified control function.

'literal'
Is a literal string to which input data is to be compared. The compare is done before the input is translated to upper case. If '*literal*' is specified, DATA must be specified in the first operand. If the input data length is not equal to the *literal* string length, the compare is performed with the smaller length, unless the conditional relationship is ¬ and the data length is zero, in which case the control function is performed. If the input is in lowercase, the ALPHA statement should be used and the literal coded in lowercase.

data-length
Specifies an integer value to which the number of characters of input data for the field is compared.

ENDMPPI—END MULTIPLE PAGE INPUT
Specifies the end of multiple physical page input (this input is the last for the message being created).

label
Specifies that testing is to continue with the IF statement bearing the label (branch). The label must be placed on an IF statement that follows the current statement in the TABLE definition (that is, it must be a forward branch function).

NEXTLP—NEXT LOGICAL PAGE
Specifies a request for the next logical page of the current message.

NEXTMSG—MESSAGE ADVANCE
Specifies a request to dequeue the output message in progress (if any) and to send the next output message in the queue (if any).

NEXTMSGP—MESSAGE ADVANCE PROTECT
Specifies a request to dequeue the output message in progress (if any), and either send the next output message or return an information message indicating that no next message exists.

NEXTPP—PAGE ADVANCE
Specifies a request for the next physical page in the current output message. If no output message is in progress, no explicit response is made.

NOFUNC
Specifies that conditional function testing is to be terminated.

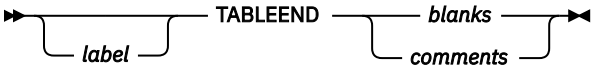
PAGEREQ—LOGICAL PAGE REQUEST
Specifies that the second through last characters of input data are to be considered as a logical page request.

TABLEEND statement

The TABLEEND statement establishes the end of a table definition.

If this is the end of the input to SYSIN processing, the TABLEEND statement must be followed by an END compilation statement.

Format



Parameters

label
A one- to eight-character alphanumeric name can be specified. It is not used.

Compilation statements

Compilation statements include the ALPHA statement, the COPY statement, the EQU statement, the RESCAN statement, the STACK statement, the UNSTACK statement, the TITLE statement, the PRINT statement, the SPACE statement, the EJECT statement, and the END statement.

ALPHA statement

The ALPHA statement specifies a set of characters to be considered alphabetic by the MFS Language utility for the purpose of defining valid field names and literals.

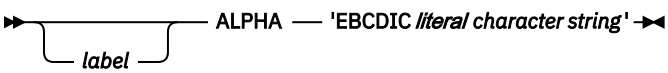
Restriction: The following characters cannot be made alphabetic using ALPHA.

b ¢ * < (+ !! *) ; ¬ .
-/, % _ > ? :
' = ”
0 through 9

The characters A through Z, &; (X'50'), #, $, and @ are always considered alphabetic by the MFS Language utility.

All the characters referred to are known as standard characters. Therefore, all other characters are referred to as nonstandard characters.

Format



Parameters

label
A one- to eight-character alphanumeric name can be specified. It is not used.

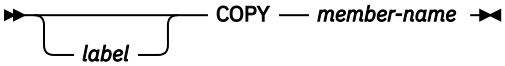
'*literal character string*'
Specifies the characters to be considered alphabetic by the MFS Language utility. The use of an EGCS literal in an ALPHA statement causes an ERROR message.

COPY statement

The COPY statement invokes a copy of a member of the partitioned data set represented by the SYSLIB DD statement.

The copied member can request the nested copy of another member. The member to be copied cannot already exist at a higher level in a nested chain of copy requests. The nesting level available for copy is limited only by the amount of storage available to the language utility preprocessor. The level of the COPY statement is indicated to the right of each printed COPY record.

Format



Parameters

label
A one- to eight-character alphanumeric name can be specified. It is not used.

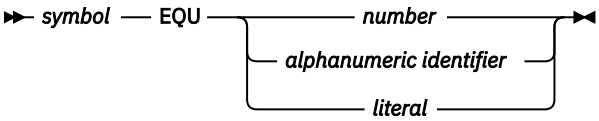
member-name
Specifies the name of the partitioned data set member to be copied into the input stream of the utility preprocessor.

EQU statement

The EQU statement defines a symbol as a substitution variable.

All subsequent occurrences of the symbol in the operand field of a statement is replaced by the value specified in the operand field of the EQU statement.

Format



Parameters

symbol
Specifies the symbol to be equated to the value specified in the operand field. The symbol must be a one- to eight-character alphanumeric identifier, the first character of which must be alphabetic.

number
Specifies the value to be represented by the symbol, and consists of 1 to 256 decimal digits.

alphanumeric identifier
Specifies the value to be represented by the symbol, and consists of 1 to 256 alphanumeric characters, the first of which must be alphabetic.

literal
Specifies the value to be represented by the symbol, and consists of 1 to 256 valid characters (not counting embedded second quotes), enclosed in quotes. The characters within the leading and trailing quotes replace the symbol when substitution occurs. An EGCS literal cannot be equated if any hexadecimal value within the literal is a X'7D' (a single quote character).

A symbol used in an equate (EQU) statement can be re-equated to another value.

There are no reserved words that cannot be used as symbols on the EQU statement. However, when defining symbols do not use a symbol as one of the words used by the MFS statement operands. Otherwise, the intended function of the MFS word cannot be used.

Example: Consider the following equate statement:

```
NOPROT EQU PROT
```

Then if one DFLD specifies ATTR=NOPROT and another DFLD specifies ATTR=PROT, both DFLDs would generate the protect attribute (PROT).

Restriction: Once an MFS word is equated, it cannot be restored to its original symbol. In other words, a symbol cannot be equated to itself.

Concatenated EQU statements

A period (.) can be used to concatenate two equated values, or one value and specific data, if a delimiter exists at the point of concatenation.

Example EQU statements

Consider the following EQU statements:

```
A  EQU ATTR
AE EQU 'ATTR='
P  EQU '(PROT,NUM)'
EP EQU '=(PROT,NUM)'
```

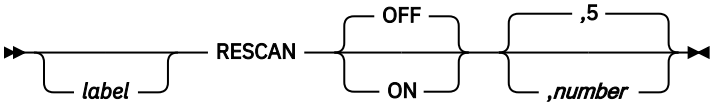
The following examples generate the same results:

```
ATTR=(PROT,NUM)
ATTR=P
AE.P
A.EP
A=P
```

RESCAN statement

The RESCAN statement controls the operation of EQU statements during replacement mode.

Format



Parameters

label
A 1- to 8-character alphanumeric name can be specified. It is not used.

OFF | ON
Specifies whether (ON) or not (OFF) replacement text should be rescanned for further substitution. The default is OFF unless a number is specified.

If ON is specified, replacement text can invoke further substitution within the substituted text up to a maximum number of occurrences.

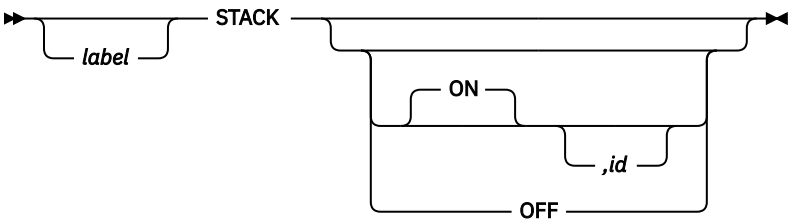
5 | *number*
Specifies how many times further substitution is allowed in a single rescan substitution. The default is 5. If more recursive substitutions are attempted than specified by the maximum *'number'*, an error message is issued and substitution terminates. RESCAN ON,0 will be interpreted as RESCAN OFF.

STACK statement

The STACK statement is used to delineate one or more SYSIN or SYSLIB records, and to request that those records, once processed, be kept (stacked) in processor storage for reuse at a later time.

A stack of SYSIN/SYSLIB records must not contain STACK and UNSTACK statements. The letter S to the right of each printed record indicates that it is being stacked for future use.

Format



Parameters

label
A 1- to 8-character name can be specified. It is not used.

ON
Specifies the beginning of a stack of SYSIN/SYSLIB records. ON is the default, and it does not have to be specified to begin stacking.

id
Specifies the one-to eight-character alphanumeric name for the record stack. If the compilation only uses one stack, no ID is required; MFS assigns an ID of eight blanks to the stack.

When multiple stacking operations are requested, all stacks should be uniquely identified; one unnamed stack is permitted.

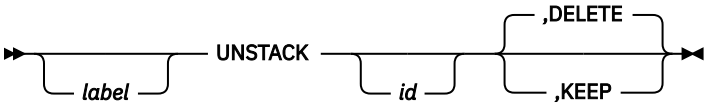
OFF
Specifies the end of a stack of SYSIN/SYSLIB records.

UNSTACK statement

The UNSTACK statement requests retrieval of a previously processed stack of SYSIN/SYSLIB records and specifies whether the retrieved stack should be deleted after processing.

The letter U to the right of each printed record indicates that it is being read from the processor storage stack for processing.

Format



Parameters

label
A one- to eight-character alphanumeric name can be specified. It is not used.

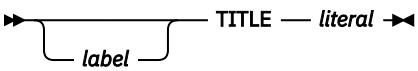
id
Specifies the 1- to 8-character identifier of the stack to be retrieved and processed. If no ID is specified, MFS retrieves the stack identified by eight blanks.

DELETE|KEEP
Specifies whether (KEEP) or not (DELETE) the stack should be retained after retrieval and processing. The default is DELETE.

TITLE statement

The TITLE statement is used to specify the heading to appear on the SYSPRINT listing.

Format



Parameters

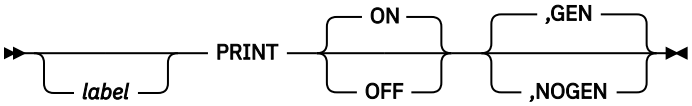
label
A one- to eight-character alphanumeric name can be specified. It is not used.

literal
Specifies the heading to be printed on the output listing. The heading can be specified as an EGCS literal. An EGCS literal of more than 108 bytes causes an error message.

PRINT statement

The PRINT statement provides printing specifications for the SYSPRINT listing.

Format



Parameters

label
A one- to eight-character alphanumeric name can be specified. It is not used.

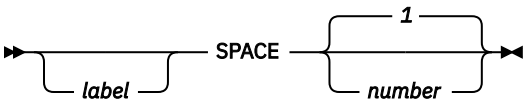
ON|OFF
Specifies whether (ON) or not (OFF) a listing should be printed. The default is ON.

GEN|NOGEN
Specifies whether (GEN) or not (NOGEN) the intermediate text blocks (ITBs) should be printed in hexadecimal following the statement at the left margin. If PRINT GEN is used following the ENDDO statement, all definitions generated for the iterative DO group are printed. The default is GEN.

SPACE statement

The SPACE statement specifies the number of lines to skip when output is printed. The SPACE statement is printed.

Format



Parameters

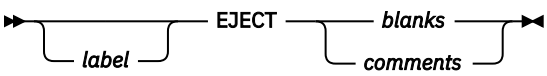
label
A one- to eight-character alphanumeric name can be specified. It is not used.

1 | ***number***
Specifies how many lines to skip after this statement is encountered. The default is 1.

EJECT statement

The EJECT statement is used to eject a page in an output listing. The EJECT statement is printed.

Format



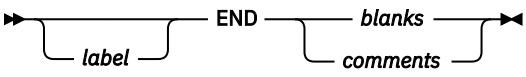
Parameters

label
A one- to eight-character alphanumeric name can be specified. It is not used.

END statement

The END statement is used to define the end of the input to SYSIN processing. If this statement is omitted, one is provided and an error message is issued.

Format



Parameters

label
A one- to eight-character alphanumeric name can be specified. It is not used.

Running the utility in standard mode

You can run the DFSUPAA0 utility in standard mode using the two-step mode of operation using the MFSUTL procedure described.

The following figure shows an overview of the two-step standard mode of operation using the MFSUTL procedure.

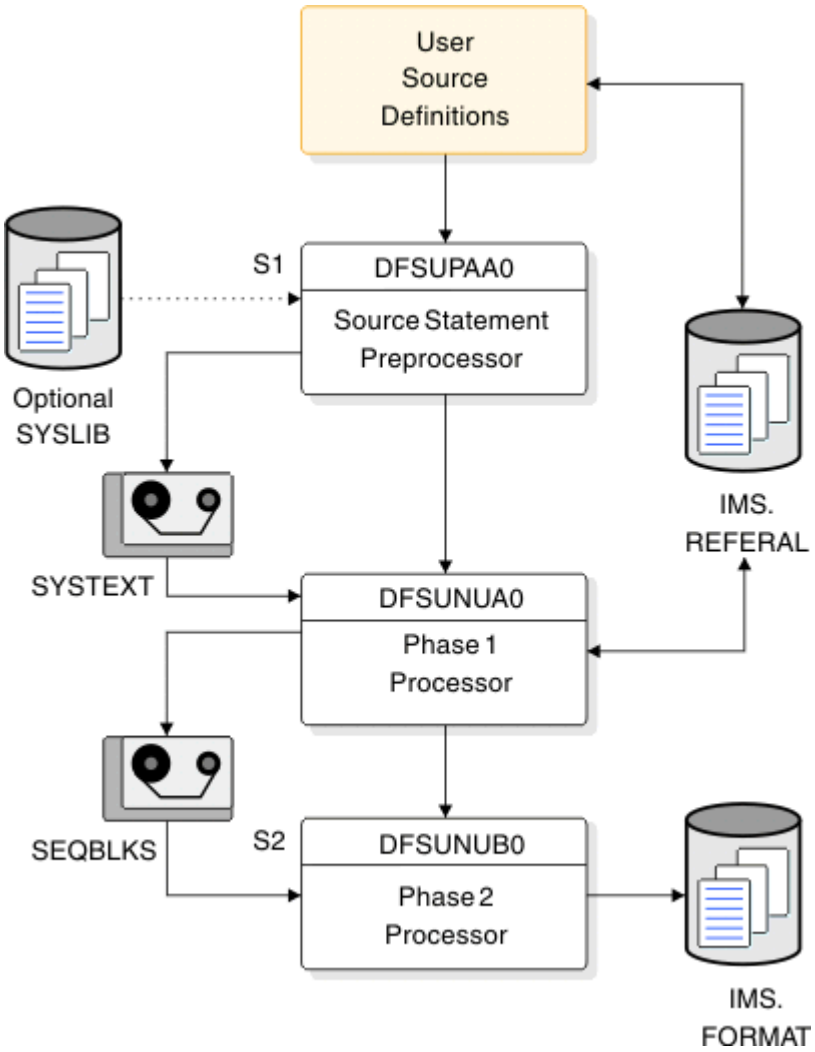


Figure 8. Overall flow with the MFSUTL procedure

The following topics discuss the details of Step 1 (S1) and Step 2 (S2).

MFSUTL procedure description

Step 1 (S1): Source statement preprocessor

The MFS Language utility preprocessor provides a nonassembler, nonmacro preprocessor for the MFS Language source statements.

The execution of the preprocessor generates intermediate text blocks (ITBs), which are then processed by the remaining utility phase to generate message (MSG) and format (FMT) control blocks. IMS uses the control blocks to format messages for device display and application program presentation.

The primary function of the preprocessor is to perform syntax and relational validity checks on user specifications. The preprocessor generates ITBs for each MSG, FMT, partition descriptor block (PDB), and TABLE source definition processed, and stores them in IMS.REFERAL.

The IMS.REFERAL partitioned data set (PDS) directory contains an entry for each MSG, FMT, PDB, and TABLE ITB. Additionally, the interrelationships between all known FMT and MSG ITBs that have been placed in the IMS.REFERAL are recorded in the PDS directory.

The preprocessor executes in the following order:

1. The preprocessor constructs a control table representing the interrelationships between all known FMT and MSG ITBs that have been placed in IMS.REFERAL. If you request the compress function in the EXEC statement, the preprocessor compresses IMS.REFERAL.
2. The preprocessor adds the user-supplied FMT and MSG definitions to the control table and resolves them against the table to ensure that, when a given definition is supplied for processing, all control blocks in the format set are reprocessed by phase 1. This resolution also allows you to compile source for only the message or format that requires change—not the entire format set.
3. After the resolution function has been accomplished to determine the definitions to be processed, the preprocessor places the control statements for these FMT and MSG definitions on the SYSTEXT data set for phase 1 processing.

If you change a PDB definition, each format set referencing that PDB might need to be recompiled.

The following actions cause phase 1 reprocessing of the format set ITBs to create new FMT and MSG control blocks for the IMS.FORMAT library:

- Modification of a FMT definition that already exists as a control block in IMS.FORMAT.
- Modification of a MSG definition that already exists as a control block in IMS.FORMAT.
- Addition of a new MSG to the format set.
- Reassignment of a MSG definition to a different FMT. This reassignment causes the old format set and the new format set to be reprocessed.

If a MSG definition refers to a FMT name (through the SOR= keyword) that has not yet been supplied to the MFS Language utility, the preprocessor stores the MSG ITB in the IMS.REFERAL library. The MSG control block is not created until a FMT definition is supplied. The format set is then processed to create the new MSG and FMT control blocks for IMS.FORMAT.

Similarly, if a FMT definition is supplied to the MFS Language utility and no MSGs refer to it, the FMT ITB is stored in IMS.REFERAL. The FMT control block is not created until at least one MSG definition is supplied. The format set is then processed to create the new MSG and FMT control blocks for IMS.FORMAT.

Each IMS error message is accompanied by a return code of 4, 8, 12, 16, or 20 to indicate increasing severity of error.

If an error condition is detected during the processing of statements that would create an FMT, MSG, PDB, or TABLE ITB, the highest such severity code associated with the message stating the error is kept and used to determine if the ITB is to be written to the IMS.REFERAL library. The preprocessor maintains the highest return code issued for each definition processed. You can specify a compare value (the lowest unacceptable return code) in the STOPRC parameter of the EXEC statement. If the return code is greater than or equal to the STOPRC value, the ITB is not written in IMS.REFERAL. For example, a STOPRC of 4 permits only ITBs that have a return code of 0 to be written. If no STOPRC is specified, a value of 8 is assumed, and only ITBs having a return code of 0 or 4 are written.

The preprocessor also maintains the highest return code for all ITBs processed during a job. Phase 1 is not given control by the preprocessor if the highest return code is greater than or equal to 16, or if no ITBs were written in IMS.REFERAL. If the return code is 16 or greater, the preprocessor returns control to z/OS with a completion code equal to the return code.

Step 1 (S1): Phase 1 processor

The preprocessor invokes the phase 1 processor. Initially, the phase 1 processor uses the control statements placed by the preprocessor on the SYSTEXT data set to construct a module table representing all of the FMT and MSG ITBs to be processed in this run. After constructing the module table, the phase 1 processor reads in ITBs from IMS.REFERAL and builds control blocks for each MSG and FMT definition. If

a TABLE of control functions is requested by an input format definition, the phase 1 processor obtains the TABLE ITB from IMS.REFERAL and builds functions into the device input format (DIF).

When a format definition requests a HALDB partition set, the phase 1 processor gets the PDB ITB from IMS.REFERAL and builds the partitioning control functions into the device output format (DOF).

The phase 1 processor places the newly constructed control blocks on the SEQBLKS data set. Each member processed has a control record placed on the SEQBLKS data set identifying the member, its size, and the date and time of creation. This control record is followed by the image of the control block as constructed by the phase 1 processor.

If an error is detected during control block building, an error control record is placed on the SEQBLKS data set for the definition in error, identifying the member in error, and the date and time the error control record was created. In addition, the phase 1 processor returns a completion code of 12 to z/OS. If execution of step 2 is forced, the phase 2 processor deletes control blocks with build errors.

The phase 1 processor maintains a high return code for all ITBs processed during an execution of the MFS Language utility. Before returning to z/OS, the phase 1 processor compares its high return code to the preprocessor's high return code. The highest of the two is passed to z/OS as the completion code for step 1.

Step 2 (S2): Phase 2 processor

The phase 2 processor receives control as a job step when the phase 1 processor is finished. The phase 2 processor operates in a two-pass mode to place the new control blocks into the IMS.FORMAT library. On the first pass, the phase 2 processor reads the SEQBLKS data set and creates an internal table that contains the name of every MOD, MID, DOF, and DIF created by the phase 1 processor. The name of the format or message description that had build errors during the phase 1 processor's execution is also added to this internal table. Control blocks in the IMS.FORMAT library that are to be replaced in IMS.FORMAT, or had build errors during phase 1, are deleted from IMS.FORMAT.

If you request the compress function, the phase 2 processor compresses the IMS.FORMAT library. This ensures maximum available library space for adding control block members and, due to the reprocessing of all related members by the phase 1 processor, allows the grouping of related control blocks for seek time reductions when fetching the control blocks during online execution.

In the second pass, the SEQBLKS data set is reprocessed, together with the module table, to write the new control blocks into IMS.FORMAT and STOW them for a directory update.

If control blocks with entries in the main-storage index directory, $$IMSDIR, were deleted and not replaced, the index entries should be deleted. This update can be done automatically, but it is inefficient for a large format library with a relatively small number of blocks deleted. To avoid this, the following parameter can be used for both the MFSUTL and MFSBTCH2 procedures: DIRUPDT=UPDATE|NOUPDATE. The default is UPDATE. If UPDATE is specified or defaulted, $$IMSDIR is updated automatically. If NOUPDATE is specified, updating is bypassed; and you must delete the blocks from $$IMSDIR with the MFS Service Utilities

If index entries are to be added to $$IMSDIR for new control blocks created in this run, the INDEX function of the MFS service utility must be used.

The phase 2 processor passes a completion code to z/OS for step 2 based on all the control block maintenance to IMS.FORMAT for a given execution of the MFS Language utility.

JCL requirements

The following figure shows the JCL for the MFSUTL procedure.

```
//         PROC RGN=4M,SOUT=A,SYS2=,
//            SNODE='IMS',
//            SOR=NOLIB,MBR=NOMBR,PXREF=NOXREF,
//            PCOMP=NOCOMP,PSUBS=NOSUBS,PDIAG=NODIAG,
//            COMPR=NOCOMPRESS,COMPR2=COMPRESS,
```

```
//            LN=55,SN=8,DEVCHAR=0,COMPR3=NOCOMPREND,
//            DIRUPDT=UPDATE,
//            NODE1=IMS,
//            NODE2=IMS
//S1       EXEC PGM=DFSUPAA0,REGION=&RGN,
//            PARM=(&PXREF,&PCOMP,&PSUBS,&PDIAG,
//            &COMPR,'LINECNT=&LN,STOPRC=&SN',
//            'DEVCHAR=&DEVCHAR')
//STEPLIB  DD DSN=&NODE2..&SYS2.SDFSRESL,DISP=SHR
//*SYSLIB - USER OPTION
//SYSIN    DD DISP=SHR,
//         DSN=&SNODE..&SOR.(&MBR)
//REFIN    DD DSN=&NODE1..&SYS2.REFERAL,DISP=OLD
//REFOUT   DD DSN=&NODE1..&SYS2.REFERAL,DISP=OLD
//REFRD    DD DSN=&NODE1..&SYS2.REFERAL,DISP=OLD
//SYSTEXT  DD DSN=&&TXTPASS,UNIT=SYSDA,
//         SPACE=(CYL,(1,1)),DCB=BLKSIZE=800
//SYSUT3   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//DUMMY    DD DISP=SHR,
//         DSN=&NODE2..&SYS2.PROCLIB(REFCPY)
//UTPRINT  DD SYSOUT=&SOUT
//SYSPRINT DD SYSOUT=&SOUT,
//         DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYSUDUMP DD SYSOUT=&SOUT&SOUT
//SEQBLKS  DD DSN=&&BLKS,DISP=(NEW,PASS),
//         UNIT=SYSDA,SPACE=(CYL,(1,1))
//S2       EXEC PGM=DFSUNUB0,REGION=&RGN,
//            PARM=(&COMPR2,&COMPR3,&DIRUPDT,
//            'DEVCHAR=&DEVCHAR'),COND=(8,LT,S1)
//STEPLIB  DD DSN=&NODE2..&SYS2.SDFSRESL,DISP=SHR
//SEQBLKS  DD DSN=&&BLKS,DISP=(OLD,DELETE)
//UTPRINT  DD SYSOUT=&SOUT,
//         DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYSUDUMP DD SYSOUT=&SOUT
//FORMAT   DD DSN=&NODE1..&SYS2.FORMAT,DISP=SHR
//DUMMY    DD DISP=SHR,
//         DSN=&NODE2..&SYS2.PROCLIB(FMTCPY)
//SYSPRINT DD SYSOUT=&SOUT
//SYSUT3   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
```

The DISP=OLD specifications are required.

Restriction: A DD DUMMY specification is not supported.

REFCPY control statement

The MFSUTL procedure uses this control statement to compress REFERAL.

```
COPY  INDD=REFOUT,OUTDD=REFOUT
```

FMTCPY control statement

The MFSUTL procedure uses this control statement to compress FORMAT.

```
COPY  INDD=FORMAT,OUTDD=FORMAT
```

Running the utility in batch mode

You can run the DFSUPAA0 utility in batch mode using the two required procedures: MFSBTCH1 and MFSBTCH2.

Batch mode provides the ability to batch the message descriptors and device formats into an accumulation data set, IMS.MFSBATCH. This data set can then be applied to the MFS staging library, IMS.FORMAT, with a separate job. The batch accumulation data set requires you to allocate and catalog an IMS system data set, IMS.MFSBATCH, large enough to hold all the control blocks that are to be accumulated before they are placed into IMS.FORMAT. The following figure shows an overview of the batch mode.

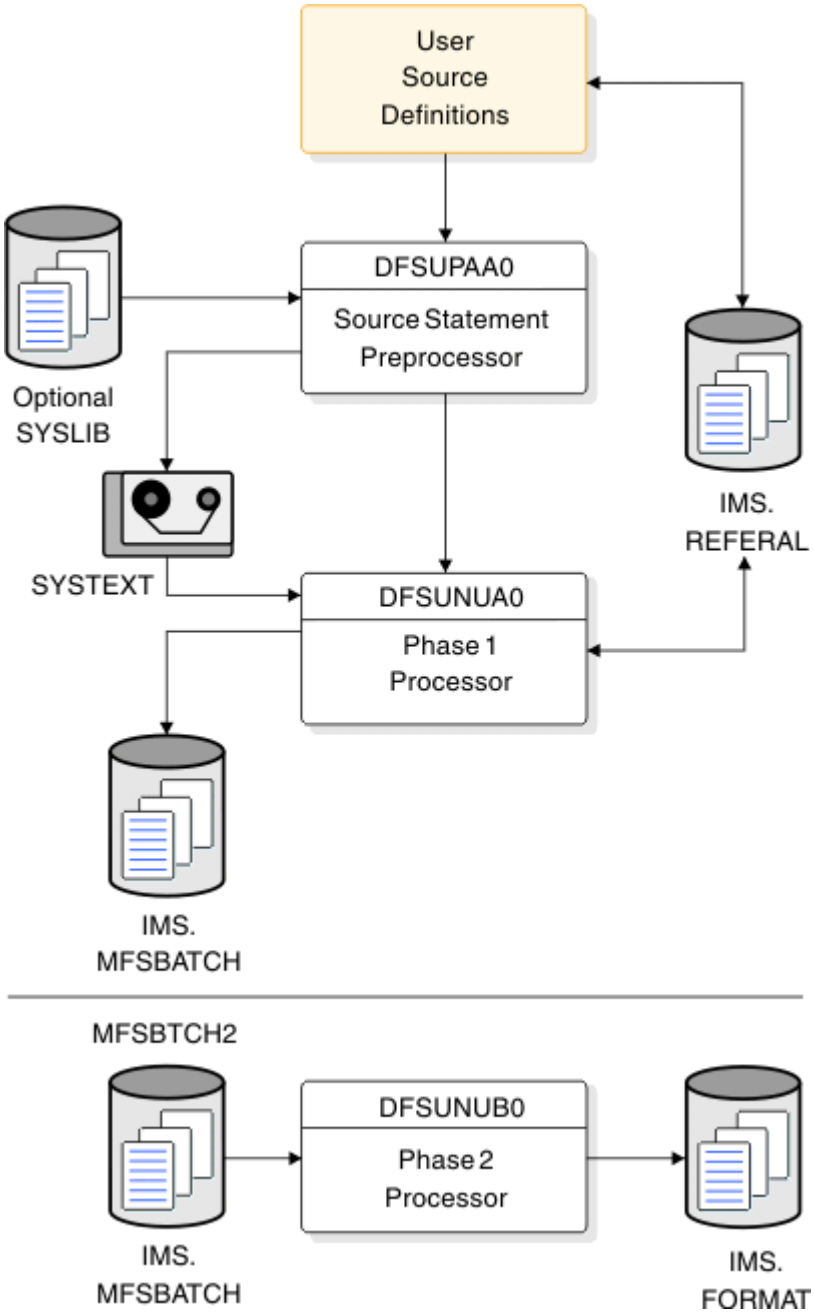


Figure 9. Overall flow with the MFSBTCH1 and MFSBTCH2 procedures

MFSBTCH1 procedure description

This procedure is identical to step 1 of the "MFSUTL procedure description" on page 251, except that the newly constructed control blocks or error control records, or both, are added to the SEQBLKS accumulation data set, IMS.MFSBATCH.

JCL requirements

The JCL for the MFSBTCH1 procedure is shown in the following example. Refer to Chapter 4, "MFS Language utility (DFSUPAA0)," on page 167 for details on the EXEC parameters and DD statements.

```
//         PROC RGN=4M,SOUT=A,SYS2=,
//            SNODE='IMS',
//            SOR=NOLIB,MBR=NOMBR,PXREF=NOXREF,
//            PCOMP=NOCOMP,PSUBS=NOSUBS,PDIAG=NODIAG,
//            COMPR=NOCOMPRESS,LN=55,SN=8,DEVCHAR=0,
```

```
//            NODE1=IMS,
//            NODE2=IMS
//S1       EXEC PGM=DFSUPAA0,REGION=&RGN,
//            PARM=(&PXREF,&PCOMP,&PSUBS,&PDIAG,
//            &COMPR, 'LINECNT=&LN,STOPRC=&SN',
//            'DEVCHAR=&DEVCHAR')
//STEPLIB  DD DSN=&NODE2..&SYS2.SDFSRESL,DISP=SHR
//*SYSLIB - USER OPTION
//SYSIN    DD DISP=SHR
//            DSN=&SNODE..&SOR.(&MBR),
//REFIN    DD DSN=&NODE1..&SYS2.REFERAL,DISP=OLD
//REFOUT   DD DSN=&NODE1..&SYS2.REFERAL,DISP=OLD
//REFRD    DD DSN=&NODE1..&SYS2.REFERAL,DISP=OLD
//SYSTEXT  DD DSN=&&TXTPASS,UNIT=SYSDA,
//            SPACE=(CYL,(1,1)),DCB=BLKSIZE=800
//SYSUT3   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//DUMMY    DD DISP=SHR
//         DSN=&NODE2..&SYS2.PROCLIB(REFCPY),
//UTPRINT  DD SYSOUT=&SOUT
//SYSPRINT DD SYSOUT=&SOUT,
//         DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYSUDUMP DD SYSOUT=&SOUT
//SEQBLKS  DD DISP=(MOD,KEEP)
//         DSN=&NODE1..&SYS2.MFSBATCH
```

REFCPY control statement

The MFSBTCH1 procedure uses this control statement to compress REFERAL.

```
COPY  INDD=REFOUT,OUTDD=REFOUT
```

MFSBTCH2 procedure description

This procedure is identical to step 2 of the “MFSUTL procedure description” on page 251, except as noted in the following paragraphs.

If control blocks with duplicate names are found, only the last one found is recorded. This ensures that if the control block with the same name was processed more than once by the MFSBTCH1 procedure, the control block created last is added to IMS.FORMAT. If the control blocks are to be replaced in IMS.FORMAT, they are first deleted from IMS.FORMAT. Consequently, if the control block created last had build time errors, and a block with the same name existed in IMS.FORMAT, the block is deleted from IMS.FORMAT.

On the second pass, IMS.MFSBATCH is reprocessed together with the module table to write the new control blocks, and last occurrences of the duplicate control blocks, into IMS.FORMAT and STOW them for a directory update.

At the end of this step, the SEQBLKS data set is emptied for subsequent use by the MFSBTCH1 procedure.

JCL requirements

The JCL for the MFSBTCH2 procedure is shown in the following example. Refer to Chapter 4, “MFS Language utility (DFSUPAA0),” on page 167 for details on the EXEC parameters and DD statements.

```
//         PROC RGN=4M,SOUT=A,COMPR2=COMPRESS,
//            COMPR3=NOCOMPREND,DIRUPDT=UPDATE,SYS2=,
//            NODE1=IMS,
//            NODE2=IMS
//S2       EXEC PGM=DFSUNUB0,REGION=&RGN,
//            PARM='&COMPR2,&COMPR3,&DIRUPDT'
//STEPLIB  DD DSN=&NODE2..&SYS2.SDFSRESL,DISP=SHR
//SEQBLKS  DD DISP=(OLD,KEEP),
//         DSN=&NODE1..&SYS2.MFSBATCH
//UTPRINT  DD SYSOUT=&SOUT,
//         DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYSUDUMP DD SYSOUT=&SOUT
```

```
//FORMAT   DD DSN=&NODE1..&SYS2.FORMAT,DISP=SHR
//DUMMY    DD DISP=SHR,
//         DSN=&NODE2..&SYS2.PROCLIB(FMTCPY)
//SYSPRINT DD SYSOUT=&SOUT
//SYSUT3   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
```

FMTCPY control statement

The MFSBTCH2 procedure uses this control statement to compress FORMAT.

```
COPY  INDD=FORMAT,OUTDD=FORMAT
```

Running the utility in test mode

You can run the DFSUPAA0 utility in test mode of operation using the MFSTEST procedure described.

The following figure shows an overview of the test mode of operation using the MFSTEST procedure.

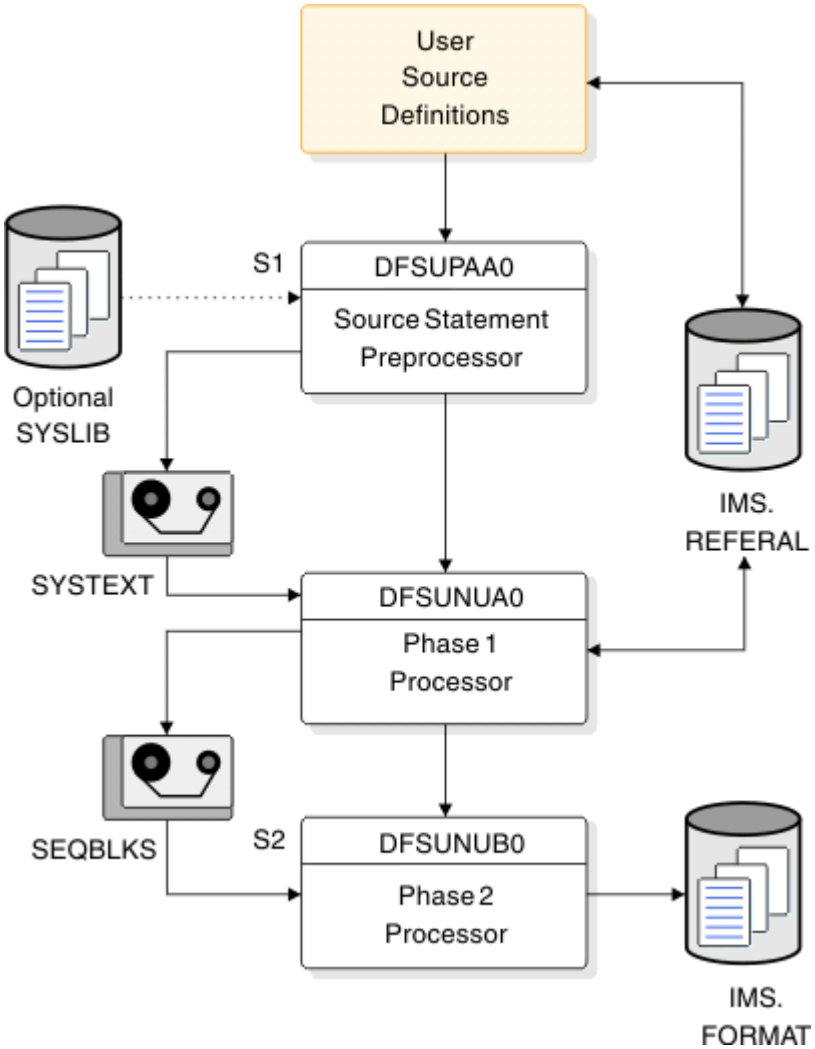


Figure 10. Overall flow with the MFSTEST procedure

MFSTEST procedure description

The MFSTEST procedure can be used to create message descriptors and device formats while an IMS control region is active. You can check the control blocks created by MFSTEST, without disrupting online production activity, by using the **/TEST MFS** command. Control blocks that have been tested can be

placed into the staging library using the MFSUTL procedure. The MFSTEST procedure does not alter the staging library, and all of the ITBs and control blocks remain stable.

The two-step execution of the MFS Language utility describes differs when the MFSTEST procedure is used.

Step 1 (S1): Source statement preprocessor

The source statement preprocessor operates in the same manner as the MFSUTL procedure with the exception that the ITBs are placed into a temporary library. The contents of the historical reference library, IMS.REFERAL, are not changed to reflect new MSG, FMT, PDB, or TABLE ITBs, or new relationships that result from this test mode execution. The IMS.REFERAL library is used only in a read-only manner to perform the resolution function that ensures that all required MSG and FMT ITBs are processed.

Step 1 (S1): Phase 1 processor

The phase 1 processor operates identically to the operation with the MFSUTL procedure, except that the ITBs for the required MSGs and FMTs are read in from the concatenated temporary library created by the preprocessor and from the IMS.REFERAL library.

The phase 1 processor obtains all MSGs, FMTs, PDBs, and TABLEs defined by this execution from the temporary library created by the preprocessor. Additional blocks, if related ITBs are present, are obtained from IMS.REFERAL.

Step 2 (S2): Phase 2 processor

The phase 2 processor operates against a special format library, IMS.TFORMAT, which is used by the IMS control region to access MFS control blocks when terminals are in MFSTEST mode. The phase 2 processor deletes control blocks from this library if new versions are created during this execution or if errors are detected during this execution. The phase 2 processor then inserts the new control blocks created during this execution into the library which will be available for online testing.

IMS.TFORMAT is not compressed, since the IMS control region might be concurrently reading from it.

Recommendation: Periodically compress this data set when the IMS control region is not executing (use DISP=OLD for IMS.TFORMAT).

The test procedure deletes $$IMSDIR, if one exists on the test format data set.

JCL requirements

The following figure shows the JCL for the MFSTEST procedure.

```
//         PROC RGN=4M,SOUT=A,SYS2=,
//            SNODE='IMS',
//            SOR=NOLIB,MBR=NOMBR,PXREF=NOXREF,
//            PCOMP=NOCOMP,PSUBS=NOSUBS,PDIAG=NODIAG,
//            COMPR=NOCOMPRESS,LN=55,SN=8,DEVCHAR=0,
//            NODE1=IMS,
//            NODE2=IMS
//S1       EXEC PGM=DFSUPAA0,REGION=&RGN,
//            PARM=(&PXREF,&PCOMP,&PSUBS,&PDIAG,
//            &COMPR,'LINECNT=&LN,STOPRC=&SN',
//            'DEVCHAR=&DEVCHAR')
//STEPLIB  DD DSN=&NODE2..&SYS2.SDFSRESL,DISP=SHR
//*SYSLIB - USER OPTION
//SYSIN    DD DISP=SHR,
//         DSN=&SNODE..&SOR.(&MBR)
//REFIN    DD DSN=&NODE1..&SYS2.REFERAL,DISP=OLD
//REFOUT   DD DSN=&&TEMPPDS,
//         DCB=&NODE1..&SYS2.REFERAL,
//         UNIT=SYSDA,SPACE=(CYL,(5,1,10))
//REFRD    DD DSN=*.REFOUT,VOL=REF=*.REFOUT,
//         DISP=(OLD,DELETE)
```

```
//         DD DSN=&NODE1..&SYS2.REFERAL,DISP=OLD
//SYSTEXT  DD DSN=&&TXTPASS,UNIT=SYSDA,
//         SPACE=(CYL,(1,1)),DCB=BLKSIZE=800
//SYSPRINT DD SYSOUT=&SOUT,
//         DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYSUDUMP DD SYSOUT=&SOUT
//SEQBLKS  DD DSN=&&BLKS,DISP=(NEW,PASS),
//         UNIT=SYSDA,SPACE=(CYL,(1,1))
//S2       EXEC PGM=DFSUNUB0,REGION=&RGN,
//             PARM='TEST,DEVCHAR=&DEVCHAR',
//             COND=(8,LT,S1)
//STEPLIB  DD DSN=&NODE2..&SYS2.SDFSRESL,DISP=SHR
//SEQBLKS  DD DSN=&&BLKS,DISP=(OLD,DELETE)
//UTPRINT  DD SYSOUT=&SOUT,
//         DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYSUDUMP DD SYSOUT=&SOUT
//FORMAT   DD DSN=&NODE1..&SYS2.TFORMAT,DISP=SHR
```

The DISP=OLD specifications are required.

Restriction: A DD DUMMY specification is not supported.

MFS library backup procedure

The MFSBACK procedure performs utility library backup operations.

Attention: When you use this procedure, make sure that the IMS.REFERAL and IMS.FORMAT libraries are dumped and restored at the same level, that is, at the same time. It is important to do this because of the relational information in the IMS.REFERAL PDS directory which describes the contents of the libraries. To ensure that all libraries are restored to the same level, scratch and reallocate all MFS data sets prior to performing the restore operation. If the libraries are not restored to the same level, unpredictable operation can occur.

MFSBACK procedure

The following figure shows the JCL for the MFSBACK procedure and includes the optional MFSTEST facility. All DISP=OLD specifications are required.

Restriction: A DD DUMMY specification is not supported in the statements that require DISP=OLD.

The block size for the IMS.REFERAL library, if specified, must be 800 bytes.

```
//       PROC  NODE='IMS',
//             TAPE=MFSDBS,SOUT=A,DSN=FORMAT,SYS2=
//*
//*****************************************************
//*                                                   *
//*  PROCEDURE KEYWORDS FOR // EXEC STATEMENT:        *
//*                                                   *
//*    NODE=   PREFIX LEVEL TO BE USED FOR            *
//*            ACCESS TO IMS MFS LIBRARIES.           *
//*                                                   *
//*    SYS2=   SECOND PREFIX LEVEL TO BE USER FOR     *
//*            ACCESS TO IMS MFS LIBRARIES.           *
//*                                                   *
//*    TAPE=   BACKUP TAPE SERIAL NUMBER.             *
//*                                                   *
//*    SOUT=   SPECIFIES THE PRINT OUTPUT CLASS       *
//*            TO BE USED FOR PRINTED OUTPUT          *
//*            DURING THE BACKUP OPERATION.           *
//*                                                   *
//*****************************************************
//*
//MOVE1  EXEC PGM=IEBCOPY,PARM='SIZE=100K'
//SYSPRINT DD SYSOUT=&SOUT
//SYSUT3   DD SPACE=(CYL,(1,1)),UNIT=SYSDA
//REFERAL  DD DSN=&NODE..&SYS2.REFERAL,DISP=OLD
//TAPEOUT  DD UNIT=2400,LABEL=(1,SL),DISP=(NEW,PASS),
//            VOL=(,RETAIN,SER=&TAPE),
//            DSN=&NODE..&SYS2.REFERAL,
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//*
```

```
//*************************************************
//*                                               *
//*  //MOVE1.SYSIN  DD  *  MUST BE SUPPLIED BY THE *
//*  USER WITH THE APPROPRIATE COPY CONTROL       *
//*  STATEMENT AS SHOWN BELOW:                    *
//*                                               *
//*   COPY OUTDD=TAPEOUT,INDD=REFERAL             *
//*                                               *
//*************************************************
//*
//MOVE2  EXEC PGM=IEBCOPY,PARM='SIZE=100K'
//SYSPRINT DD SYSOUT=&SOUT
//SYSUT3   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//FORMAT   DD DSN=&NODE..&SYS2.&DSN,DISP=OLD
//TAPEOUT  DD UNIT=2400,LABEL=(2,SL),
//            VOL=(,RETAIN,REF=*.MOVE1.TAPEOUT),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
//            DSN=&NODE..&SYS2.&DSN,
//            DISP=(OLD,KEEP)
//*
//*************************************************
//*
//*  //MOVE2.SYSIN  DD  *  MUST BE SUPPLIED WITH   *
//*  APPROPRIATE COPY CONTROL STATEMENT           *
//*  AS SHOWN BELOW:                              *
//*                                               *
//*    COPY OUTDD=TAPEOUT,INDD=FORMAT             *
//*                                               *
//*************************************************
//*       //MOVE3  EXEC PGM=IEBCOPY,PARM='SIZE=100K'
//SYSPRINT DD SYSOUT=&SOUT
//SYSUT3   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//FORMAT   DD DSN=&NODE..&SYS2.TFORMAT,DISP=SHR
//TAPEOUT  DD UNIT=2400,LABEL=(3,SL),
//            VOL=REF=*.MOVE1.TAPEOUT,
//            DSN=&NODE..&SYS2.TFORMAT,
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
//            DISP=(OLD,KEEP)
//*
//*************************************************
//*  //MOVE3.SYSIN  DD  *  MUST BE SUPPLIED WITH   *
//*  THE APPROPRIATE COPY CONTROL STATEMENT       *
//*  AS SHOWN BELOW:                              *
//*                                               *
//*   COPY OUTDD=TAPEOUT,INDD=FORMAT              *
//*                                               *
//*************************************************
//*
```

MFS restore procedure

The MFSREST procedure performs utility restore operations.

Attention: When you use this procedure, make sure that the IMS.REFERAL and IMS.FORMAT libraries are dumped and restored at the same level, that is, at the same time. It is important to do this because of the relational information in the IMS.REFERAL PDS directory which describes the contents of the libraries. To ensure that all libraries are restored to the same level, scratch and reallocate all MFS data sets prior to performing the restore operation. If the libraries are not restored to the same level, unpredictable operation can occur.

MFSREST procedure

The following figure shows the JCL for the MFSREST procedure and includes the optional MFSTEST facility. All DISP=OLD specifications are required.

Restriction: A DD DUMMY specification is not supported in the statements that require DISP=OLD.

```
//       PROC  NODE='IMS',
//             TAPE=MFSDBS,SOUT=A,DSN=FORMAT,SYS2=
//*
//*************************************************
//*                                               *
//*  PROCEDURE KEYWORDS FOR // EXEC STATEMENT:    *
```

```
//*                                                     *
//*    NODE=   PREFIX LEVEL TO BE USED FOR              *
//*            ACCESS TO IMS MFS LIBRARIES.             *
//*                                                     *
//*    SYS2=   SECOND PREFIX LEVEL TO BE USED FOR       *
//*            ACCESS TO IMS MFS LIBRARIES.             *
//*                                                     *
//*    TAPE=   RESTORE TAPE SERIAL NUMBER.              *
//*                                                     *
//*    SOUT=   SPECIFIES THE PRINT OUTPUT CLASS         *
//*            TO BE USED FOR PRINTED OUTPUT            *
//*            DURING THE RESTORE OPERATION.            *
//*                                                     *
//*******************************************************
//*
//MOVE1  EXEC PGM=IEBCOPY,PARM='SIZE=100K'
//SYSPRINT DD SYSOUT=&SOUT
//SYSUT3   DD SPACE=(CYL,(1,1)),UNIT=SYSDA
//REFERAL  DD DSN=&NODE..&SYS2.REFERAL,DISP=OLD
//TAPEIN   DD UNIT=2400,LABEL=(1,SL),DISP=(OLD,KEEP),
//            VOL=(,RETAIN,SER=&TAPE),
//            DSN=&NODE..&SYS2.REFERAL,
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//*
//*******************************************************
//*                                                     *
//*  //MOVE1.SYSIN  DD  *  MUST BE SUPPLIED BY THE     *
//*  USER WITH THE APPROPRIATE COPY CONTROL             *
//*  STATEMENT AS SHOWN BELOW:                          *
//*                                                     *
//*   COPY OUTDD=REFERAL,INDD=TAPEIN                    *
//*                                                     *
//*******************************************************
//*
//MOVE2  EXEC PGM=IEBCOPY,PARM='SIZE=100K'
//SYSPRINT DD SYSOUT=&SOUT
//SYSUT3   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//FORMAT   DD DSN=&NODE..&SYS2.&DSN,DISP=OLD
//TAPEIN   DD UNIT=2400,LABEL=(2,SL),
//            VOL=(,RETAIN,REF=*.MOVE1.TAPEIN),
//            DSN=&NODE..&SYS2.&DSN,
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
//            DISP=(OLD,KEEP)
//*
//*******************************************************
//*  //MOVE2.SYSIN  DD  *  MUST BE SUPPLIED WITH        *
//*  THE APPROPRIATE COPY CONTROL STATEMENT             *
//*  AS SHOWN BELOW:                                    *
//*                                                     *
//*    COPY OUTDD=FORMAT,INDD=TAPEIN                    *
//*                                                     *
//*******************************************************
//*
//MOVE3  EXEC PGM=IEBCOPY,PARM='SIZE=100K'
//SYSPRINT DD SYSOUT=&SOUT
//SYSUT3   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//FORMAT   DD DSN=&NODE..&SYS1.TFORMAT,DISP=SHR
//TAPEIN   DD UNIT=2400,LABEL=(3,SL),
//            VOL=REF=*.MOVE1.TAPEIN,
//            DSN=&NODE..&SYS2.TFORMAT,
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
//            DISP=(OLD,KEEP)
//*
//*******************************************************
//*  //MOVE3.SYSIN  DD  *  MUST BE SUPPLIED WITH        *
//*  THE APPROPRIATE COPY CONTROL STATEMENT             *
//*  AS SHOWN BELOW:                                    *
//*                                                     *
//*   COPY OUTDD=FORMAT,INDD=TAPEIN                     *
//*                                                     *
//*******************************************************
//*
```

Chapter 5. Program Specification Block (PSB) Generation utility

The Program Specification Block (PSB) Generation utility generates PSBs from macro instructions that describe the characteristics of application programs and the use of logical terminals and logical data structures.

Unless the IMS management of ACBs is enabled in an IMS system, the PSB Generation utility is required before you can run an application program under IMS. The ACB Maintenance utility uses the PSBs together with *database descriptors* (DBDs) to create *application control blocks* (ACBs) for use during run time.

When the IMS management of ACBs is enabled in an IMS system, instead of using the PSB Generation utility, you can use SQL Dynamic Definition Language (DDL) statements, such as **CREATE PROGRAMVIEW**, to describe the characteristics of an application program to IMS.

The PSB generation statements supply the identification and characteristics of the IMS resources to be used. These *program communication blocks* (PCBs) represent message destinations and databases that are used by the application program. In addition, there must be a statement that supplies the characteristics of the application program itself. There must be one PSB for each message, batch, or Fast Path program. The name of the PSB and its associated application program must be the same in a telecommunications system.

If you require only an I/O PCB and a single, modifiable alternate PCB, you can use a generated PSB (GPSB) to describe the resources that are required for your application program. GPSBs can be used in any online environment, and are typically used in DCCTL application programs. You do not need to perform PSBGEN for GPSBs.

Attention: If the PROCOPT values allow a BMP application to insert, replace, or delete segments in the databases, ensure that a BMP application does not update a combined total of more than 300 databases and HALDB partitions without committing the changes.

All full-function database types that have uncommitted updates count against the limit of 300. When you design a HALDB database, use particular caution because the number of partitions in HALDB databases is the most common reason for approaching the 300 database limit for uncommitted updates.

Subsections:

- “Restrictions” on page 263
- “Prerequisites” on page 263
- “Requirements” on page 264
- “Recommendations” on page 264
- “Input and output” on page 264
- “JCL specifications” on page 265

Restrictions

Currently, no restrictions are documented for the PSB Generation utility.

Prerequisites

Currently, no prerequisites are documented for the PSB Generation utility.

© Copyright IBM Corp. 1974, 2024

Requirements

Currently, no requirements are documented for the PSB Generation utility.

Recommendations

Currently, no recommendations are documented for the PSB Generation utility.

Input and output

The PSB Generation utility places the created PSB in the PSB library. Each PSB is a member of the operating system partitioned data set IMS.PSBLIB. For IMS batch execution (DL/I region type), the necessary database PCB PSB is loaded from PSBLIB and the expanded PSB needed for DL/I database PCB statement processing is built from it. ACB generation must be performed to prebuild the expanded PSBs into the ACBLIB. PSBLIB is used as input to the ACB generation process. Batch executions can also use prebuilt blocks from the ACBLIB by specifying region type 'DBB' on the JCL execute statement. When an application that is running in an online region (BMP) references a PSB with one or more GSAM PCBs defined, IMS uses ACBLIB with PSBLIB to build its internal control blocks. In this case, the PSB must be defined the same in both ACBLIB and PSBLIB.

The six types of statements used for a PSB generation are:

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

The list of statements used for a PSB generation does not include a PCB for the input message source. I/O PCBs exist within the IMS online control program nucleus for this purpose. Upon entry to the application program used for message processing, a PCB pointer to the source of the input message is provided as the first entry in a list of PCB address pointers. The remainder of the PCB list has a direct relationship to the PCBs as defined within the associated PSB and must be defined in the application program in the same order as defined during PSB generation. All PCBs can be used by the application program when making DL/I message and database calls. Only one PCB is used in a particular DL/I call.

You can exclude alternate, DL/I, Fast Path, and GSAM PCBs from the PCB list that is passed to the application program by defining a name for the PCB (PCBNAME=name) and specifying LIST=NO. You must name the PCB when you want to issue calls using the application interface block (AIB). The AIB can be used for all types of PCBs.

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

In the case of a batch program, no I/O PCB exists in the list unless you request it with the CMPAT option on the PSBGEN statement. Therefore, if CMPAT=YES is not specified, the PCB list provided to the program

has a direct relationship to the PCBs within the PSB. No TP PCBs should be contained in a PSB for batch processing in a batch processing region.

In a TM batch environment, CMPAT=YES is implied and cannot be overridden by PSBGEN. The PCB list for application programs running in a DCCTL batch region always contains an I/O PCB.

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

You can specify alternate and modifiable alternate PCBs in a PSB associated with a Fast Path program executing in a Fast Path region. A response alternate PCB with the same PTERM can be used to send a Fast Path output message back to the original PTERM with a different component attached to the terminal. You can use an alternate PCB (non-response mode) to send an output message to any terminal or IMS message queue.

You can reference the PCB list passed to the application program upon entry to the application program by the names defined within the application program for making DL/I calls and interrogating PCB information (status codes and feedback information). The address of a PCB can be the second parameter in a DL/I call from an application program to IMS. The PCB address can represent the source of the input message, the destination for an output message, or a database. Upon completion of a DL/I call, the PCB contains status and feedback information pertinent to the call.

Output messages and statistics

PSB generation produces three types of printed output and one load module, which becomes a member of the partitioned data set, IMS.PSBLIB. The types of output are:

Control Statement Listing
This is a listing of the input statement images to this job step.

Diagnostics
Errors discovered during the processing of control statement result in diagnostic messages being printed immediately following the image of the last control statement read before the error was discovered. The message can either refer to the control statement immediately preceding it or the preceding group of control statements. It is also possible for more than one message to be printed for each control statement. In this case, they follow each other on the output listing. After all the control statements have been read, a further check is made of the logic of the entire deck. This can result in one or more additional diagnostic messages.

If an error is discovered, a diagnostic message is printed, the control statements are listed, and the other outputs are suppressed. However, all the control statements are read and checked before the PSB generation execution is terminated. The bind step of PSB generation is not executed if a control statement error has been found.

Assembler Listing
Except when PRINT NOGEN is specified, an operating system assembler language listing of the PSB created by PSB generation execution is provided.

Load Module
PSB generation is a two-step operating system job. Step 1 is a macro assembly execution that produces an object module. Step 2 is a bind of the object module, which produces a load module that becomes a member of IMS.PSBLIB.

JCL specifications

The PSB Generation utility executes as a standard operating system job. You must define an EXEC statement and an Utility control statement.

EXEC statement

Must be in the format

```
// EXEC PSBGEN,MBR=APPLPGM1
```

Related concepts

Building the application control blocks (ACBGEN) (Database Administration)

Allocating ACBLIB data sets (System Definition)

Utility control statements

Utility control statements define the required and optional statements for the PSBGEN utility.

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

Alternate PCB statement

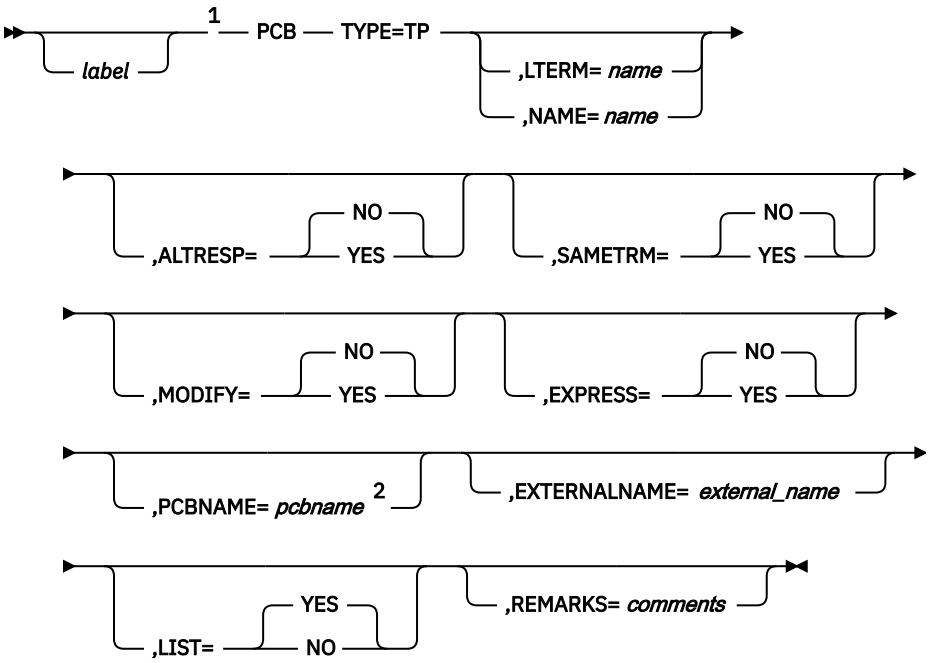
The alternate PCB describes a destination other than the source of the current input message.

This statement instruction allows the application program to send output messages to a destination other than the source of an input message.

Requirement: A PCB statement is required for each destination to which output is to be sent.

These messages can be sent to either an output terminal or an input transaction queue to be processed by another program. Each output message destination requires a separate alternate PCB destination. If the input source terminal is all that is required to respond with output, do not include any PCB statements of this type. Message processing programs, batch message processing programs, and Fast Path programs can have alternate PCB statements in their associated PSBs. An alternate PCB cannot be used to send a message to a Fast Path transaction; however, Fast Path application programs can use an alternate PCB to route messages to any terminal or IMS transaction.

Alternate PCB statements must be first in the PSB generation control card deck, followed by the statements identifying PCBs associated with IMS databases. The following diagram shows the alternate PCB statement format.



Notes:

1 *label* and PCBNAME are mutually exclusive. Use either the label or the PCBNAME= parameter.

2 *label* and PCBNAME are mutually exclusive. Use either the label or the PCBNAME= parameter.

label

Specifies an alphanumeric label from 1 to 8 characters long, that is valid for an assembler language statement. The labels for the PCB statement within a PSB must be unique.

Exception: Do not specify this parameter if the PCBNAME= parameter is used.

PCB

Indicates that this is a PCB statement.

TYPE=TP

Is a required keyword parameter for all alternate PCBs.

LTERM=|NAME=

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

EXTERNALNAME=

An optional alias for the PCB label or the PCBNAME= parameter. Java application programs use the external name to refer to the PCB.

Specify an external name as a 1- to 128-character uppercase alphanumeric string. An external name can include underscore characters.

The external name must be unique within a PSB.

When EXTERNALNAME is not specified, the default external name is either the PCB label or the PCBNAME, whichever has been specified.

If no PCB label or PCBNAME is specified, EXTERNALNAME defaults to blanks.

Restriction: External names cannot be reserved SQL keywords or begin with DFS.

If the EXTERNALNAME parameter is not specified and a reserved SQL keyword is specified in the NAME parameter, EXTERNALNAME accepts the NAME value as the default external name after appending "_SCH" to the NAME value.

ALTRESP=
Specifies whether (YES) or not (NO) this alternate PCB can be used instead of the I/O PCB for responding to terminals in response mode, conversational mode, or exclusive mode. The default value is NO. ALTRESP=YES is only valid for alternate PCBs.

SAMETRM=
Specifies whether (YES) or not (NO) IMS verifies that the logical terminal named in the response alternate PCB is assigned to the same physical terminal as the logical terminal that originated the input message. The default value is NO. You must specify SAMETRM=YES for response alternate PCBs used by conversational programs and programs operating with terminals in response mode. SAMETRM=NO should be specified if alternate response PCBs are used to send messages to output-only devices that are in exclusive mode.

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

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

YES
When specified, indicates EXPRESS messages can be sent to the destination terminal even though the program abends or issues a ROLL or ROLB call. For all PCBs (express or non-express) under these conditions, messages inserted but not made available for transmission are canceled, while messages made available for transmission are never canceled.

For a non-express PCB, the message is not available for transmission to its destination until the program reaches a sync (commit) point. The sync point occurs when the program terminates, issues a CHKP call, or requests the next input message (if the transaction is defined with MODE=SNGL).

For an express PCB, the message is available for transmission to the destination when IMS knows it has the complete message. The message is available when a PURG call is made using that PCB, or when the program requests the next input message.

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

NO
When specified, indicates messages are backed out if the application program abends. NO is the default.

PCBNAME=
Specifies the name of the PCB. The PCB name must be an alphanumeric, 8-byte character string that follows standard naming conventions. The PCB name must be unique within the PSB.

Exception: Do not specify this parameter if a label is used.

LIST=
Specifies whether the named PCB is included in the PCB list passed to the application program at entry. Specify YES to include a named PCB in the PCB list. Specify NO to exclude a named PCB from the PCB list. YES is the default.

To exclude a PCB from the PCB list, you must assign the PCB a name with the PCBNAME= parameter. You can specify LIST=NO if an application program does not need a PCB's address.

REMARKS=
Optional user comments. A 1- to 256-character field.

If your comments contain special characters, such as commas or blank spaces, enclose the full comment string in single quotation marks.

The value specified on the REMARKS keyword cannot contain the following characters:

- Single quotation marks, except when they are used to enclose the full comment string. If a single quotation mark is entered before the end of the full comment string, the remainder of the comment string is truncated. The following examples show correct and incorrect usages of single quotation marks on the REMARKS keyword:

  CORRECT

  ```
  REMARKS='These remarks apply to the XYZ application'
  ```

  INCORRECT

  ```
  REMARKS='These remarks apply to the 'XYZ' application'
  ```
- Double quotation marks.
- Less than (<) symbols.
- Greater than (>) symbols.
- Ampersands (&).

Full-function or Fast Path database PCB statement

The second type of statement in a PSB generation input record specifies a description of a PCB for a DL/I or a Fast Path database.

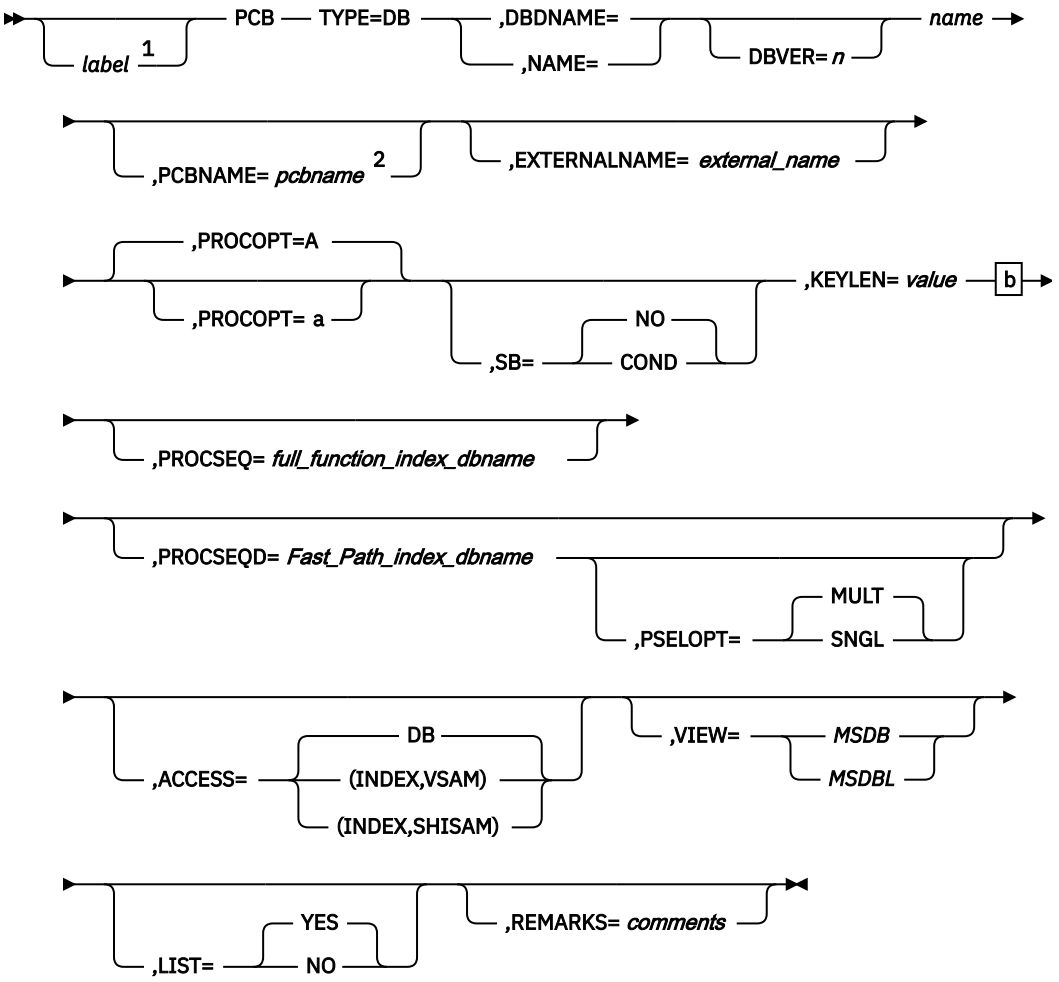
Although one or more database PCBs are usually included in a PSB, the second type of statement is not always required. For example, a message switching program or conversational message program might not require access to a DL/I database. Therefore, a database PCB is not required.

In a DCCTL environment, database PCBs (except for GSAM PCBs) are not supported, but might be included in the PSBGEN. Application programs that execute in a DCCTL environment and that attempt to use a database PCB will receive an AD status code.

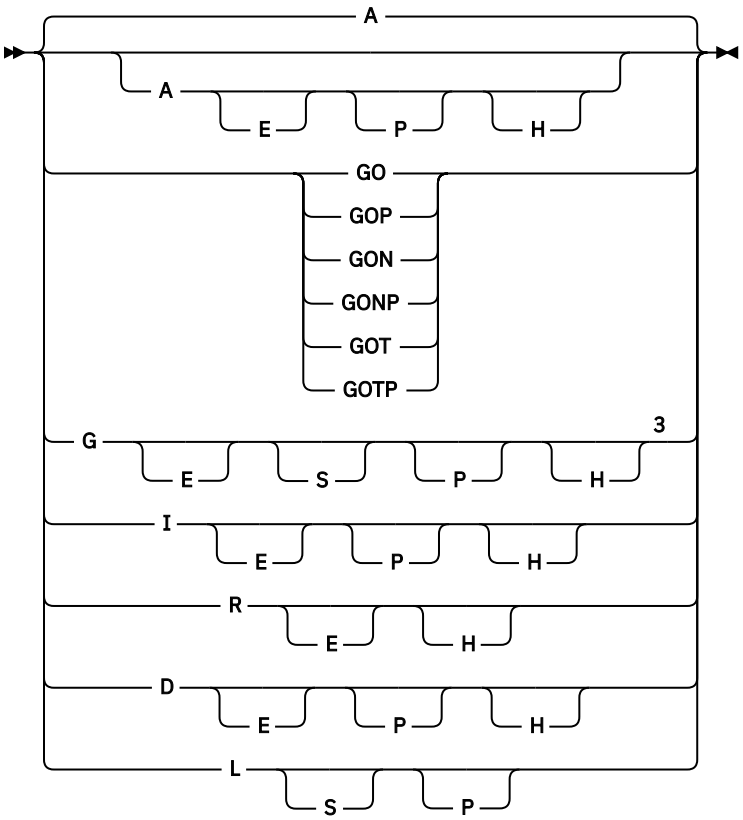
The maximum number of database PCBs that can be defined in a PSBGEN is 2500, including alternate terminal PCBs. This is the maximum value for application programs executing in all IMS region types (MSG, DL/I, and so on).

If a Fast Path secondary index PCB is the only PCB in the PSB, the associated DEDB PCB must be included in the PSB. The minimal DEDB PCB requires a SENSEG statement for the root segment of the associated DEDB database.

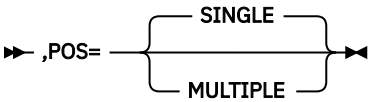
The following diagram shows the format for the DL/I database PCB statement.


label
1
PCB
TYPE=DB
,DBNAME=
,NAME=
DBVER=n
name
,PCBNAME= pcbname
2
,EXTERNALNAME= external_name
,PROCOPT=A
,PROCOPT= a
,KEYLEN= value
b
,SB=
NO
COND
,PROCSEQ= full_function_index_dbname
,PROCSEQD= Fast_Path_index_dbname
,PSELOPT=
MULT
SNGL
,ACCESS=
DB
(INDEX,VSAM)
(INDEX,SHISAM)
,VIEW=
MSDB
MSDBL
,LIST=
YES
NO
,REMARKS= comments

a



b



Notes:

[1] Both *label* and PCBNAME can be specified. If they have the same name, then PSBGEN will be successful, and that name will be used. If they do not have the same name, then PSBGEN will fail.

[2] Both *label* and PCBNAME can be specified. If they have the same name, then PSBGEN will be successful, and that name will be used. If they do not have the same name, then PSBGEN will fail.

[3] These operands can be selected in any combination; if G, I, R, and D are selected, use A instead (A = G, I, R, and D combined).

label

An optional label used to allow the SBPARM control statement in the DFSCTL file to reference specific PCBs. If specified, this must be an alphanumeric 1- to 8-byte character string that is valid for an assembler language statement. The labels for the PCB statements within a PSB must be unique.

Note: If both *label* and PCBNAME= are specified, they must have the same name, or PSBGEN will fail.

TYPE=DB

Is a required keyword parameter for all DL/I database PCBs.

DBDNAME= or NAME=

Is the parameter for the name that specifies the physical or logical DBD to be used as the primary source of database segments for this logical data structure. The logical structure, which is defined under this PCB with one or more SENSEG statements, is the hierarchical set of data segments to which the associated application program is sensitive. This logical hierarchy of data segments might or might not exist as a physical hierarchy. This depends on the relationship of segments defined by SENSEG statements and the existence of these segments in one or more databases as defined by their database descriptions (DBDs). All SENSEG statements that follow this statement and

precede the next PCB or PSBGEN statement must refer to segments defined in the DBD named in the DBDNAME= or NAME= parameter of this PCB.

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

DBVER=*n*
When database versioning is enabled, specifies the version number of the database definition (DBD) that this application program requires.

The numeric value specified must match a version number defined on the DBD and stored in the IMS catalog. Valid values for a database version number are 0 to 2147483647.

If multiple PCBs within a PSB refer to the same database, each PCB must specify the same DBD version number.

PCBNAME=
Specifies the name of the PCB. The PCB name must be an alphanumeric, 8-byte character string that follows standard naming conventions.

Note: If both *label* and PCBNAME= are specified, they must have the same name, or PSBGEN will fail.

PROCOPT=
Specifies the processing options for the sensitive segments that are declared in this PCB. You can use these specified options in an associated application program. You can use a maximum of four options with this parameter. The letters in the parameter have the following meaning:

A
All options. By default, PROCOPT=A includes the G (get), I (insert), R (replace), and D (delete) options. PROCOPT=A is the default setting.

G
Get option.

I
Insert option. By default, PROCOPT=I includes the G (get) option for Fast Path DEDBs; PROCOPT=I does not include the G option for other database types.

R
Replace option. By default, PROCOPT=R includes the G (get) option.

D
Delete option. By default, PROCOPT=D includes the G (get) option.

P
Path calls. Required if command code D is to be used, except for ISRT calls in a batch program that is not sensitive to fields. PROCOPT=P is not required if command code D is used when processing DEDBs. P is used in conjunction with the A (all), G (get), I (insert), D (delete), and L (load) options.

O
If the O option is used for a PCB, IMS does not check the ownership of the segments returned. Therefore, the read without integrity program might get a segment that has been updated by another program. If the updating program abends and backs out, the read without integrity program will have a segment that does not exist in the database and never did. If a segment has been deleted and another segment of the same type has been inserted in the same location, the segment data, and all subsequent data returned to the application, can be from a different database record. Therefore, if you use the O option, do not update based on data read with that option. O must be specified as GO, GON, GONP, GOT, GOTP, or GOP only.

IMS recognizes some of these error types and converts them to abend U0849. However, other conditions that occur under PROCOPT GOx are not detected as having been caused by the read-without-integrity. It is possible to get loops, hangs, and system abends. When using this PROCOPT, carefully consider system design to determine if concurrent update activity is likely to cause higher risk of these kinds of conditions.

N

Reduces the number of abends that read-only application programs are subject to. Read-only application programs can reference data being updated by another application program. When this happens, an invalid pointer to the data might exist. If an invalid pointer is detected, the read-only application program abends. By specifying N, you avoid this. A GG status code is returned to the program instead. The program must determine whether to terminate processing, continue processing by reading a different segment, or access the data using a different path. N must be specified as GON, GONH, or GONP.

T

Similar to the N option, except that T causes DL/I to automatically retry the operation. If the retry fails, a GG status code is returned to the application program. T must be specified as GOT, GOTH, or GOTP.

E

Enables exclusive use of the database or segment by online programs. Used in conjunction with G, I, D, R, and A.

Restriction: For a DEDB, PROCOPT=E is not permitted.

L

Load option for database loading (except HIDAM and PHIDAM).

GS

Get segments in ascending sequence only (HSAM only). If you specify GS for HSAM databases, they will be read using the Queued Sequential Access Method (QSAM) instead of the basic Sequential Access Method (BSAM) in a DL/I IMS region.

LS

Load segments in ascending sequence only (HIDAM, HDAM, PHIDAM, PHDAM). This load option is required for HIDAM and PHIDAM. Because you must specify LS for HIDAM and PHIDAM databases, the index for the root segment sequence field will be created at the time the database is loaded.

H

Specifies high-speed sequential processing for the application program that is using a particular PSB. The following restrictions apply to using PROCOPT=H:

- It can be used for DEDBs only.
- It is allowed on the PCB level and not on the segment level.
- It must be used with other Fast Path processing options.
- A maximum of four PROCOPT options can be specified, including H.
- It can only be specified for BMPs.
- Only one PROCOPT=H PCB per database per PSB is allowed. If a BMP that is using HSSP uses multiple PCBs with PROCOPT=H for the same database within the same PSB, all database calls that are using a PCB other than the first one that is used will receive an FH status code. You can use the NOPROCH keyword on the SETO statement to alleviate this restriction.
- PROCOPT=H cannot be used if PROCSEQD=*Fast_Path_index_dbdname* is specified.
- PROCOPT=H cannot be used with PROCOPT=GO.

H is used in conjunction with A, G, I, R, and D.

If you do not specify the PROCOPT parameter, it defaults to PROCOPT=A. The replace and delete functions also imply the Get function.

A user abend (U8XX) from the retrieve module (DFSDLR00) can occur with PROCOPT=GO if another program updates pointers when this program is following the pointers. A U0800 or U0852 abend can also occur in the VLEXP routine, or in the retrieve module, if an invalid compressed segment is detected. Pointers are updated during the insert and delete functions and during replacement of a variable-length segment. To reduce the number of abends of this type, code the PROCOPT= parameter with an N or a T.

Note:

1. If any PCBs in the PSB have a PROCOPT of L or LS and either explicitly reference HISAM or HIDAM databases, or implicitly reference INDEX databases, no other PCB in the same PSB can reference any of the databases listed, either explicitly or implicitly, with a PROCOPT other than L or LS. If any PCB in the PSB has a PROCOPT of L or LS and explicitly references a PHIDAM database, no other PCB in the same PSB can reference the PHIDAM database with a PROCOPT of L or LS. The SENSEG statements within that PCB should not contain INDICES= operands.
2. If L is specified for a PCB that references a database with multiple data set groups, the PCB should include at least one SENSEG statement for each data set group in the database.
3. When the first ISRT call is issued using a PCB with PROCOPT=L, and the database is using VSAM, the VSAM data set must be empty. If it is not empty, an open error will result.

   Recommendation: If the database is using OSAM, use a newly allocated empty data set.

   If the data set is not empty, the load will start at the front of the data set, writing over the existing data.
4. If the 'O' option is used for a PCB, the SENSEG statement must not specify a PROCOPT of I, R, D, or A.
5. An online application program always has exclusive use of the SHSAM or HSAM databases, which are referenced by PCBs in its PSB. No other application programs can be concurrently scheduled to access those same SHSAM or HSAM databases in an online environment.
6. If the Online Database Image Copy utility refers to this PCB, the value of PROCOPT= L or LS is invalid. If the database to be copied is the index portion of a HIDAM or PHIDAM database, only PROCOPT=G and PROCOPT=GO are valid. If PROCOPT=E is specified, the Online Image Copy utility will execute with exclusive control of the database, even though the utility does not require the control.
7. If the Database Surveyor utility feature refers to this PCB, you must specify PROCOPT=G.
8. In the case of concatenated segments, the PROCOPT= parameter governs the logical child segment of the concatenated segment. The logical parent of the concatenated segment is governed by the RULES= parameter of the SEGM statement.
9. PROCOPT=E only applies to the database specified in the PCB. To enable exclusive use of a secondary index not explicitly used by the application, add another PCB with PROCOPT=E for the secondary index database.

SB=

Specifies which PCBs will be buffered using sequential buffering (SB). This is an optional parameter. The default is SB=NO, unless the default option has been modified for Batch and BMPs by the DFSSBUX0 to SB=COND.

COND

Specifies that SB should be activated conditionally. IMS will monitor statistics about the I/O reference pattern of this PCB to the DB data set. If IMS detects a sequential I/O reference pattern and a reasonable activity rate, it will activate SB and acquire the required buffers.

NO

Specifies that SB should not be used for this DB PCB.

Tip: For short-running MPPs, Fast Path programs, and CICS® programs, either omit the SB= keyword or specify SB=NO.

KEYLEN=

The value specified in bytes of the longest concatenated key for a hierarchic path of sensitive segments that the application program uses in the logical data structure. The following figure shows an IMS database that contains segments A- H plus segment J. Segments A, B, C, D, F, and J each have a key field length of 10 bytes. Segment E has a key field length of 250 bytes. Segment G has a key field length of 40 bytes. And Segment H has a key field length of 50 bytes. The following table shows how the KEYLEN= will be specified.

Table 21. How a KEYLEN is determined

Database hierarchical paths	Concatenated key length paths
A+B+C=	30 bytes

Table 21. How a KEYLEN is determined (continued)	
Database hierarchical paths	**Concatenated key length paths**
A+B+D=	30 bytes
A+E=	260 bytes
A+F+G+H+J=	120 bytes
A KEYLEN=260 bytes would be specified	

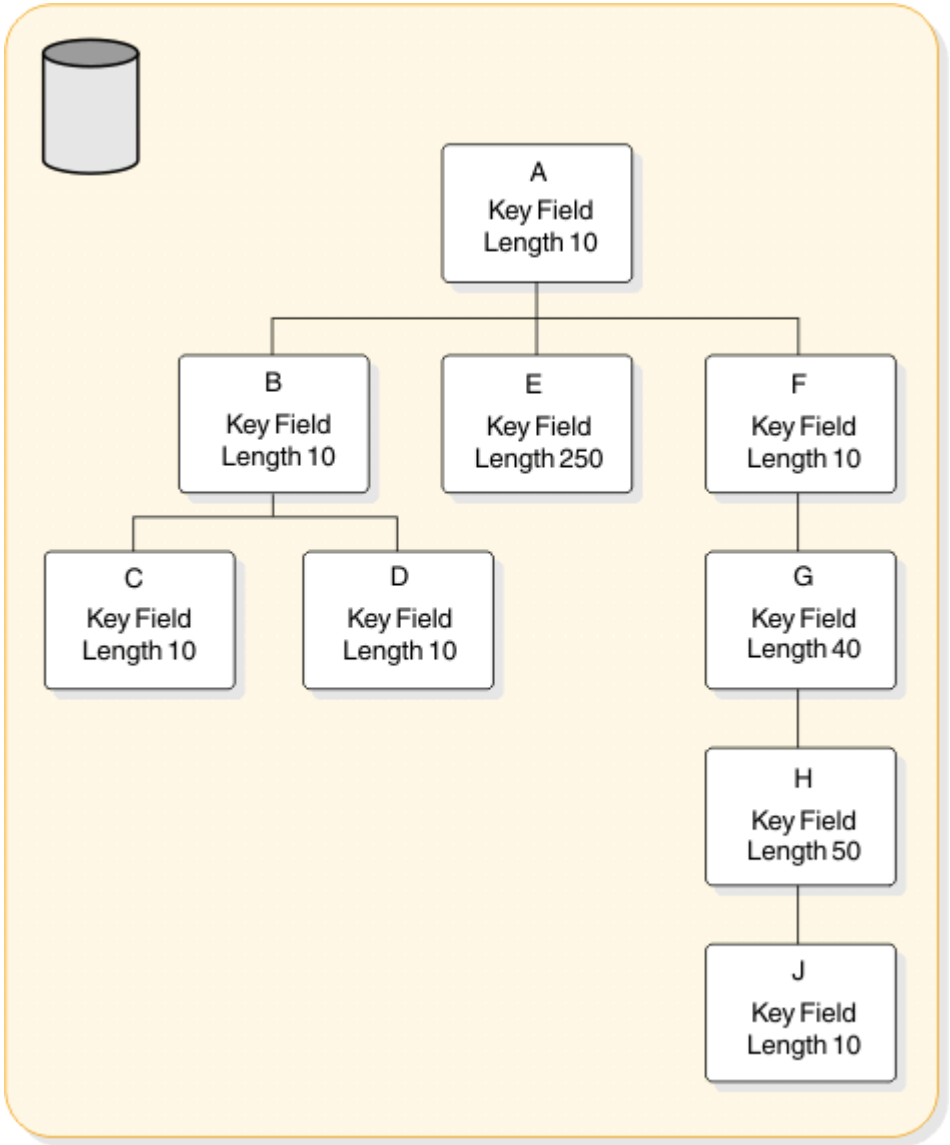


Figure 11. KEYLEN definition

For a non-terminal-related MSDB without terminal-related keys, the value must be greater than or equal to the value of the BYTES parameter of the sequence field in the DBD generation and be from 1 to 240 bytes.

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

POS=
Specifies single or multiple positioning for the logical data structure. Single or multiple positioning provides a functional variation in the call.

The performance variation between single and multiple positioning is insignificant. HSAM does not support multiple positioning.

POS=SINGLE or S is the default.

Exception: For DEDBs having more than two dependent segments, the default is POS=MULTPLE or M.

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

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

full_function_index_dbname must be the name of a secondary index DBD.

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

PROCSEQD=
Specifies the name of the secondary index database to be used to access the segments in the primary DEDB database. The parameter is optional. It is valid only if a secondary index exists for this database. If this parameter is used, subsequent SENSEG statements must be coded in the physical sequence of the primary DEDB. The minimum SENSEG statements to be coded are those in the path from the physical root segment, to the index segment (also called the target segment).

For partitioned secondary index databases, *Fast_Path_index_dbname* must be the first *dbdname* listed in the LCHILD statement.

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

When user partitioning is requested for HISAM or SHISAM secondary index databases, the PROCSEQD= parameter specifies the name of the first partition database in the user partition group as defined in the NAME= parameter on the LCHILD statement in the primary DEDB database DBD. The user partition selection exit in the PSELRTN= parameter on the XDFLD statement determines the actual partition database to use based on the secondary index key value.

PSELOPT=
Indicates how user partition databases in a user partition group are logically grouped for qualified GN calls with no SSA processing before the end of data is reached on the user partition databases. User partition databases are defined as part of a user partition group in the NAME= parameter on the LCHILD statement. This parameter applies to Fast Path secondary index databases only.

The PSELOPT= parameter can also be specified on the XDFLD statement. There is no default for the PSELOPT= parameter on the PCB statement with the PROCSEQD= parameter, whereas PSELOPT=MULT is the default for the PSELOPT= parameter on a XDFLD statement.

If the PSELOPT= parameter is specified on both the XDFLD statement and the PCB statement with the PROCSEQD operand, the PSELOPT= parameter on the PCB statement takes precedence.

MULT
Indicates the selected user partition and its subsequent user partition databases in a user data partition group as they are physically defined in the NAME= parameter on the LCHILD statement of the primary DEDB database DBD. PSELOPT=MULT is the default for the PSELOPT= parameter on a XDFLD statement.

SNGL
Indicates that only the selected user partition database is used.

ACCESS=
Specifies whether the secondary index database is to be used to access its primary DEDB database or the secondary index database is to be processed as a separate logical database.

DB
Specifies that the primary DEDB database is accessed using its secondary index sequence. ACCESS=DB is the default.

(INDEX,VSAM) | (INDEX,SHISAM)
Specifies that one or more user partition databases in a user partition group are accessed as a separate logical database.

Specify ACCESS=(INDEX,VSAM) for a HISAM secondary index database on the DBD statement for a Fast Path secondary index database.

Specify ACCESS=(INDEX,SHISAM) for a SHISAM secondary index database on the DBD statement for a Fast Path secondary index database.

VIEW=MSDB or VIEW=MSDBL

Applications can use the default DEDB commit view, or can use either the MSDB or MSDBL commit view if you specify VIEW=MSDB or VIEW=MSDBL. If you specify VIEW=MSDBL, a lock is held for the duration of DL/I data access and then released at the end of the call.

If you choose to use the default DEDB commit view after having previously specified VIEW=MSDB or VIEW=MSDBL, no changes to existing application programs are required to migrate your MSDBs to DEDBs.

If you issue a REPL call with a PCB that specifies VIEW=MSDB or VIEW=MSDBL, the segment must have a key. This includes any segment in a path if command code 'D' is specified. Otherwise, status AM is returned.

LIST=
Specifies whether the named PCB is included in the PCB list passed to the application program at entry. Specify YES to include a named PCB in the PCB list. Specify NO to exclude a named PCB from the PCB list. YES is the default.

To exclude a PCB from the PCB list, you must assign the PCB a name with either the label or PCBNAME= parameter. You can specify LIST=NO if an application program does not need a PCB's address.

EXTERNALNAME=
An optional alias for the PCB label or the PCBNAME= parameter. Java application programs use the external name to refer to the PCB.

Specify an external name as a 1- to 128-character uppercase alphanumeric string. An external name can include underscore characters.

The external name must be unique within a PSB.

When EXTERNALNAME is not specified, the default external name is either the PCB label or the PCBNAME, whichever has been specified.

If no PCB label or PCBNAME is specified, EXTERNALNAME defaults to blanks.

Restriction: External names cannot be reserved SQL keywords or begin with DFS.

If the EXTERNALNAME parameter is not specified and a reserved SQL keyword is specified in the NAME parameter, EXTERNALNAME accepts the NAME value as the default external name after appending "_SCH" to the NAME value.

REMARKS=
Optional user comments. A 1- to 256-character field.

If your comments contain special characters, such as commas or blank spaces, enclose the full comment string in single quotation marks.

The value specified on the REMARKS keyword cannot contain the following characters:

- Single quotation marks, except when they are used to enclose the full comment string. If a single quotation mark is entered before the end of the full comment string, the remainder of the comment string is truncated. The following examples show correct and incorrect usages of single quotation marks on the REMARKS keyword:

  CORRECT
  ```
  REMARKS='These remarks apply to the XYZ application'
  ```
  INCORRECT
  ```
  REMARKS='These remarks apply to the 'XYZ' application'
  ```
- Double quotation marks.
- Less than (<) symbols.
- Greater than (>) symbols.
- Ampersands (&).

Related reference
Portable SQL keywords restricted by the IMS Universal JDBC drivers (Application Programming)

Processing options for Fast Path databases

Processing options for Fast Path databases vary, depending on the Fast Path database type: non-terminal-related or fixed-terminal-related MSDB, dynamic terminal-related MSDB, or DEDB.

In a non-terminal-related or fixed terminal-related MSDB, only the processing options G and R are valid.

G
Get function.

R
Replace function. Includes G.

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

G
Get function.

I
Insert function.

R
Replace function. Includes G.

D
Delete function. Includes G.

A
All. Includes functions G, I, R and D.

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

G
Get function.

I
Insert function.

R
Replace function. Includes G.

D
Delete function. Includes G.

A

All. Includes functions G, I, R, and D.

P

Position function. Is not required if command code D is used when processing DEDBs. It is only valid for a batch message program (BMP). If this option is specified for another type of region, such as an IFP region, it will be ignored. With this option, a GC status code is returned when a UOW boundary is crossed during a G(H)U, G(H)N, or ISRT on a root segment. Also, database positioning is maintained across a valid SYNC call and a blank status code is returned when the sync is issued immediately after receiving a GC status code. In the case of a sync process failure or ROLB call, position is set to the last valid sync point or, if no valid sync point exists, to the start of the database. A SYNC or ROLB call without a preceding GC status will also cause position to be set to the start of the database.

If you use the D command code in a call to a DEDB, the P processing option need not be specified in the PCB for the program.

N

Reduces the number of abends that read-only application programs are subject to. Read-only application programs can reference data being updated by another application program. When this happens, invalid pointer to the data might exist. If an invalid pointer is detected, the read-only application program abends. By specifying N, you avoid this. A GG status code is returned to the program, instead. The program can then terminate processing, continue processing by reading a different segment, or access the data using a different path. N must be specified as GON, GONH, or GONP.

O

Read only; do not enqueue to check availability. Selecting PROCOPT=GO, GON, or GOT for DEDBs indicates that read without integrity is in effect. No locking mechanism is used to maintain the integrity of the retrieved data. O must be specified as GO, GON, or GOT, and may not be used in conjunction with H.

With O, IMS reads the control interval (CI) once and uses the same copy of the CI for the next 50 references within the same syncpoint interval. After 50 references occur, the reference counter resets and the CI will be read again. This occurs to prevent segment chain loops due to access to stale data updated by another thread.

A user abend (U1026) can occur with PROCOPT=GO if another program updates pointers when this program is following the pointers. Another example of the abend U1026 is if this program rereads a segment that has moved when another program changes its length. The following examples will help illustrate instances where abend U1026 could occur or old data is retrieved.

Example 1: If one region uses both update and PROCOPT=GO PCBs to update and read the same segment, the following scenario will not produce a pointer error to the control blocks of the PROCOPT=GO PCB (MLTE). Call the update PCB (PCBA), and the read PCB (PCBGO).

1. Region 1 PCBGO reads the CI and sets the position of the segment in MLTE. The data in the buffer is linked to EPSTGOBF.
2. Region 1 issues a call to update the segment. Region 1 PCBA steals the buffer off its EPSTGOBF. Region 1 PCBA saves the old position and updates the segment. Even if the segment is moved, Region 1 will update the PCBGO MLTE because the position in the GO MLTE matches the saved old position.
3. Region 1 PCBGO references the segment again and retrieves the updated segment.

Example 2: When two regions update the same segment and use both update and PROCOPT=GO PCBs, the following scenario will not produce a pointer error to the control blocks of the PROCOPT=GO PCB (MLTE), but the PROCOPT=GO PCB will not have access to the updated segment from the other region.

1. Region 1 PCBGO reads the CI and sets the position of the segment in MLTE. The buffer is linked to EPSTGOBF.

2. Region 2 PCBA reads the CI with lock and replaces the segment with a length change. The position of the segment changes, resulting in an FSE in the updated CI at the position set in Region 1 PCBGO MLTE. Region 1 still has the old data in the buffer which is linked to EPSTGOBF.
3. Region 1 PCBGO references the segment again and retrieves the old segment because its buffer has not been updated by Region 2's change.

Example 3: When two regions update the same segment and use both update and PROCOPT=GO PCBs, the following scenario will not produce a pointer error to the control blocks of the PROCOPT=GO PCB (MLTE), but the PROCOPT=GO PCB will not have access to the updated segment from its own region.

1. Region 1 PCBGO reads the CI and sets the position of the segment in MLTE. The buffer is linked to EPSTGOBF.
2. Region 2 PCBA reads the CI with lock and replaces the segment with a length change. The position of the segment changes, resulting in an FSE in the updated CI at the position set in Region 1 PCBGO MLTE. Region 1 still has the old data in the buffer which is linked to EPSTGOBF.
3. Region 1 issues a call to update the segment. Region 1 waits for the release of Region 2's lock. Because the updated segment is now on a different block, Region 1 does not find the duplicate buffer on EPSTGOBF and the old buffer is still linked to EPSTGOBF. Region 1 reads the update CI, which is now in its buffer. Region 1 PCBA updates the segment in its place. Even if the segment is moved, Region 1 will not update the PCBGO MLTE because the position in the MLTE no longer matches the position of the segment. There are now two duplicate buffers, one containing the old data that is linked to EPSTGOBF, and another containing updated information that is linked to EPSTXCOC.
4. Region 1 PCBGO references the segment and retrieves the old data.

Example 4: When two regions update the same segment and use both update and PROCOPT=GO PCBs, the following scenario will produce a pointer error to the control blocks of the PROCOPT=GO PCB (MLTE).

1. Region 1 PCBGO reads the CI and sets the position of the segment in MLTE. The buffer is linked to EPSTGOBF.
2. Region 2 PCBA reads the CI with lock and replaces the segment with a length change. The position of the segment changes within the same block and creates an FSE in the updated CI at the position set in Region 1 PCBGO MLTE. Region 1 still has the old data in the buffer linked to EPSTGOBF.
3. Region 1 issues a call to update the segment. Region 1 waits for the release of Region 2's lock. Region 1 PCBA steals the buffer off EPSTGOBF and reads the updated CI, moving it to Region 1's buffer. Region 1 PCBA updates the segment in its place. Even if the segment is moved, Region 1 will not update the PCBGO MLTE because the position in the MLTE no longer matches the position of the segment.
4. Region 1 PCBGO references the segment again and receives abend U1026 since there is now an FSE where the segment had been (MLTE's position).

To reduce the number of abends of this type, code the PROCOPT= parameter with an N or a T.

T
Works exactly like the N option. T must be specified as GOT, GOTH, or GOTP.

H
HSSP. Includes G and P.

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

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

The processing option of P is valid only when specified for a root segment to be used by an IMS batch message program. If the processing option P is specified for another type of region, such as an IFP region, it will be ignored. With this option, a GC status code is returned when a UOW boundary is crossed during a G(H)U, G(H)N, or ISRT on a root segment. Also, database positioning is maintained across a valid SYNC call and a blank status code is returned when the sync is issued immediately after receiving a GC status code. In the case of a sync process failure or ROLB call, position is set to the last valid sync point or, if no valid sync point exists, to the start of the database. A SYNC or ROLB call without a preceding GC status will also cause position to be set to the start of the database.

If you use the D command code in a call to a DEDB, the P processing option need not be specified in the PCB for the program.

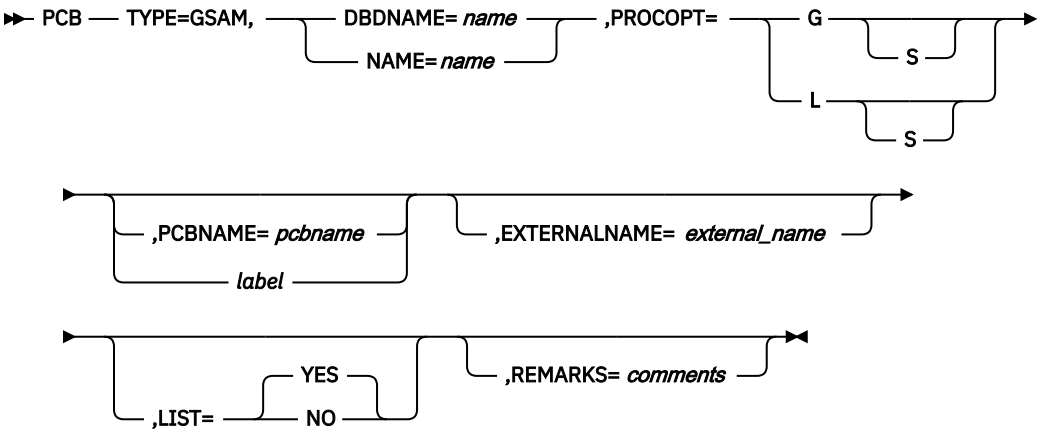
Procopt H may not be used in conjunction with O.

If you specify invalid processing options, the PSBGEN accepts them but the Application Control Blocks Maintenance utility fails. The error does not appear in the PSBGEN but appears in the ACBGEN.

GSAM PCB statement

The GSAM PCB statement specifies the name of the GSAM database PCB that will be allocated.

The following diagram shows the format for the GSAM database PCB statement.



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

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

PROCOPT=
Is a required parameter for the processing options on the data set declared in this PCB that can be used in an associated application program. Use the following characters to specify the parameter.

G
Get function.

L
Load function.

S
Large-scale sequential activity. Use GSAM multiple-buffering option (BUFFIO).

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

TP PCBs
First

DB PCBs
Second

GSAM PCBs
Last

PCBNAME=
Specifies the name of the PCB. The PCB name must be an alphanumeric, 8-byte character string that follows standard naming conventions. The PCB name must be unique within the PSB.

Exception: Do not specify this parameter if the PCB statement includes *label*.

label
Specifies an 1- to 8-character alphanumeric label that is valid for an assembler language statement. The labels for the PCB statements within a PSB must be unique.

Exception: Do not specify this parameter if PCBNAME= is used.

EXTERNALNAME=
An optional alias for the PCB label or the PCBNAME= parameter. Java application programs use the external name to refer to the PCB.

Specify an external name as a 1- to 128-character uppercase alphanumeric string. An external name can include underscore characters.

The external name must be unique within a PSB.

When EXTERNALNAME is not specified, the default external name is either the PCB label or the PCBNAME, whichever has been specified.

If no PCB label or PCBNAME is specified, EXTERNALNAME defaults to blanks.

Restriction: External names cannot be reserved SQL keywords or begin with DFS.

If the EXTERNALNAME parameter is not specified and a reserved SQL keyword is specified in the NAME parameter, EXTERNALNAME accepts the NAME value as the default external name after appending "_SCH" to the NAME value.

LIST=
Specifies whether the named PCB is included in the PCB list passed to the application program at entry. Specify YES to include a named PCB in the PCB list. Specify NO to exclude a named PCB from the PCB list. YES is the default.

To exclude a PCB from the PCB list, you must assign the PCB a name with the PCBNAME= parameter. You can specify LIST=NO if an application program does not need a PCB's address.

REMARKS=
Optional user comments. A 1- to 256-character field.

If your comments contain special characters, such as commas or blank spaces, enclose the full comment string in single quotation marks.

The value specified on the REMARKS keyword cannot contain the following characters:

- Single quotation marks, except when they are used to enclose the full comment string. If a single quotation mark is entered before the end of the full comment string, the remainder of the comment string is truncated. The following examples show correct and incorrect usages of single quotation marks on the REMARKS keyword:

  CORRECT
  ```
  REMARKS='These remarks apply to the XYZ application'
  ```
  INCORRECT
  ```
  REMARKS='These remarks apply to the 'XYZ' application'
  ```
- Double quotation marks.
- Less than (<) symbols.
- Greater than (>) symbols.

- Ampersands (&).

SENSEG statement

You use the SENSEG statement with the database PCB statement to define a hierarchically related set of data segments.

This set represents segments to which a program through this PCB is sensitive. This segment set can physically exist in one database or can be derived from several physical databases. One or more SENSEG PCB statements can be included. Each SENSEG statement must immediately follow the PCB statement to which it is related. There must be one SENSEG statement for each segment to which the application program is sensitive. All segments in the hierarchic path to any required segment must be specified. A maximum of 30,000 SENSEG statements can be defined in a single PSB generation. 30,000 SENSEG statements are impractical because this many SENSEG statements will require more storage than is usually available.

The order in which SENSEG statements are sequenced after a PCB statement determines the logical access order for the segments. When using HSAM or HISAM databases, the SENSEG statement sequence must follow the physical sequence of the segments as defined in DBDGEN, unless the PROCSEQ parameter is used in the PCB statement.

If the PROCSEQ parameter is used in the PCB statement, the SENSEG statement sequence reflects the secondary processing sequence specified by the PROCSEQ parameter. For HDAM, HIDAM, PHDAM, and PHIDAM databases, the SENSEG statements for segments on the same level do not have to be in the same order as the DBD. The order of dependent segments whose parent segment does not use hierarchic pointing can differ from the physical sequence.

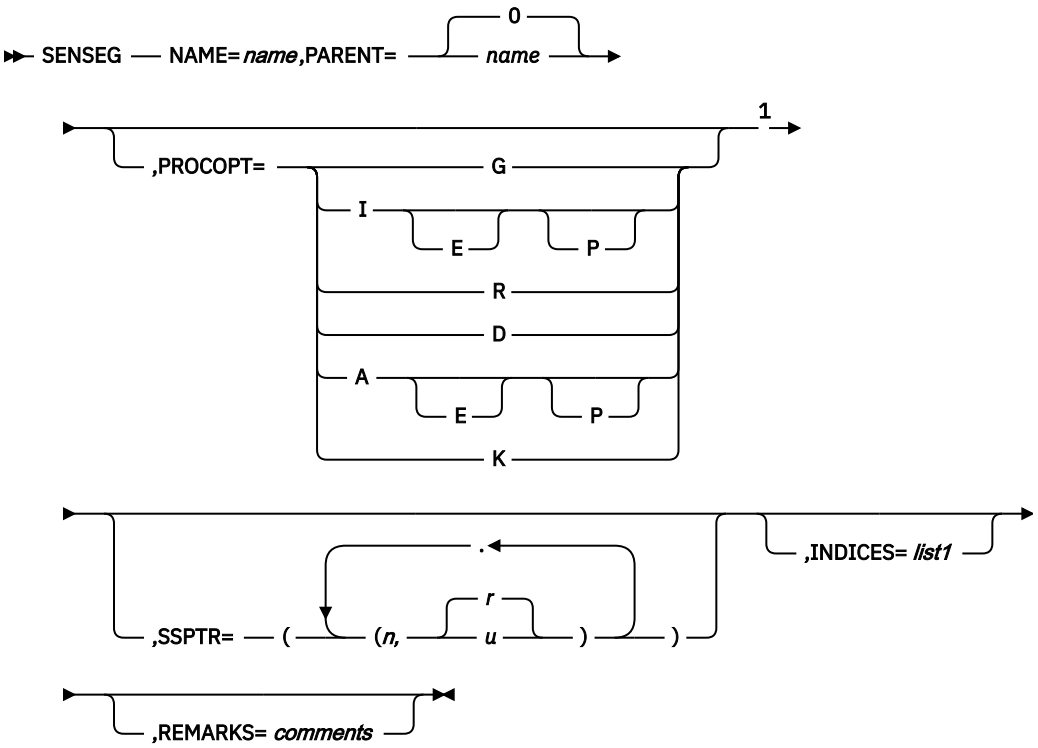
When the PROCSEQD parameter is specified on a PCB statement, the SENSEG statements must be specified using the physical structure order of the primary DEDB database. For every PCB with the PROCSEQD operand specified, it is counted as two PCBs toward the 2500 PCB limit per PSBGEN.

If the target segment is a root segment, you specify the SENSEG segments the same way as you would specify for a DEDB database without a secondary index. All segments under the root segment are accessible in the entire physical structure of the primary DEDB database using a secondary index.

If the target segment is not a root segment, you must specify SENSEG statements for all segments that are the direct parents of the target segment along the physical path from the root segment to the target segment. Only the segments that are the direct parents of the target segment along the physical path from the root segment, and all children segments of the target segment are accessible in the physical structure of the primary DEDB database when the primary DEDB database is accessed using a secondary index.

For a DEDB database, when the PROCSEQD parameter is specified on a PCB statement and the target segment is not the root segment, every SENSEG statement in the direct line from the physical root segment to the target segment must be coded. The order of the SENSEG statements must also be in physical order, even though the PROCSEQD GN processing navigates in logical order, starting from target segment going up to the root.

The format of the SENSEG statement is as follows:



Notes:

[1] These can be selected in any combination; if G, I, R, and D are all chosen, use A instead (A = G, I, R, and D combined).

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

PARENT=
Is the segment type name of this segment's parent.

Requirement: This parameter is required for all dependent segments.

The field is either from 1- to 8-alphanumeric characters or 0. If this SENSEG statement defines a root segment type as being sensitive, this parameter must equal zero. PARENT=0 is the default.

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

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

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

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

It is not valid to code the N or T processing option in the SENSEG statement. You can code them only in the PCB statement.

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

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

SSPTR=

Specifies the subset pointer number and the sensitivity for the pointer. Up to 8 subset pointers can be defined. The subset pointer number (the first parameter) must be 1 through 8. The sensitivity for the pointer (the second parameter) must be R (read sensitive) or U (update). If the first parameter and the second parameter are not specified, the pointer has no sensitivity. If only n is specified, the pointer is read sensitive. SSPTR=R is the default.

You cannot use U (update sensitivity) if the processing option is not A, R, I, or D.

INDICES=

Specifies which secondary indexes contain search fields that are used to qualify SSAs for an indexed segment type. The INDICES= parameter can be specified for indexed segment types only. It enables SSAs of calls for the indexed segment type to be qualified on the search field of the index segment type contained in each secondary index specified.

Restriction:

- An SSA of a call for an indexed segment type cannot be qualified on the search field of a secondary index unless that secondary index was specified in the INDICES= parameter of the SENSEG statement for the indexed segment type or in the PROCSEQ= parameter of the PCB statement.
- The INDICES= parameter is not supported for Fast Path secondary indexing.

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

REMARKS=

Optional user comments. A 1- to 256-character field.

If your comments contain special characters, such as commas or blank spaces, enclose the full comment string in single quotation marks.

The value specified on the REMARKS keyword cannot contain the following characters:

- Single quotation marks, except when they are used to enclose the full comment string. If a single quotation mark is entered before the end of the full comment string, the remainder of the comment string is truncated. The following examples show correct and incorrect usages of single quotation marks on the REMARKS keyword:

  CORRECT

  ```
  REMARKS='These remarks apply to the XYZ application'
  ```

  INCORRECT

  ```
  REMARKS='These remarks apply to the 'XYZ' application'
  ```

- Double quotation marks.
- Less than (<) symbols.
- Greater than (>) symbols.
- Ampersands (&).

The following figure shows the data structure of segment definition and includes segments A- F.

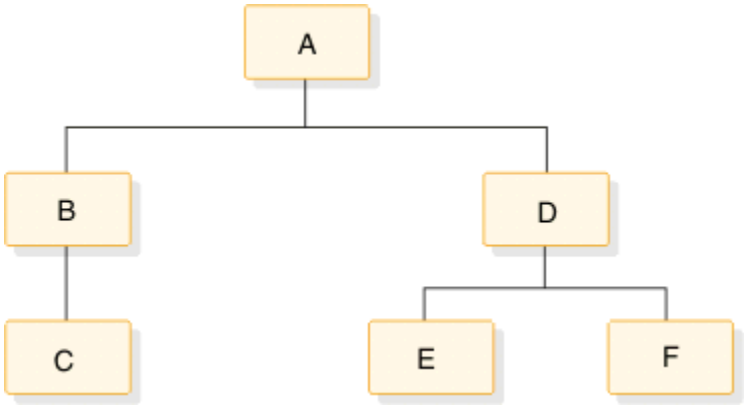


Figure 12. Data structure of segment definition

All of these segments are defined within one DBD. Do not specify INDICES= on a SENSEG PCB statement if you specified PROCOPT=L, LS, I, or D on the preceding PCB statement.

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

```
Col. 10         Col. 16                          72.

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

SENFLD statement

The SENFLD statement is used with the SENSEG statement to indicate those fields within a segment to which an application program is sensitive.

One or more SENFLD statements can be included. Each statement must follow the SENSEG statement to which it is related. You can define a maximum of 255 SENFLD statements for a given SENSEG statement. You can define a maximum of 10 000 SENFLD statements in a single PSB generation.

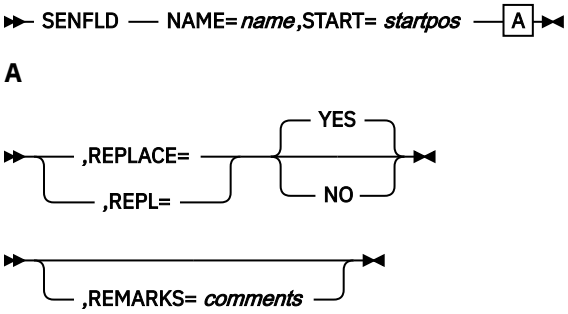
The same field can be referenced in more than one SENFLD statement within a SENSEG. If the duplicate field names participate in a concatenated segment and the same field name appears in both portions of the concatenation, the first reference will be to the logical child, and all subsequent references will be to the logical parent. This referencing sequence determines the order in which fields will be moved to the user's I/O area.

For retrieve-only processing you can request, using the SENFLD statement, that the same data be moved to multiple locations in your I/O area, provided that no overlapping occurs, and that SENFLDs of variable-length segments are of the same type.

The following restrictions apply to the SENFLD statement:

- The length field of a variable-length segment cannot be referenced through a SENFLD statement.
- A SENFLD statement cannot appear within a SENSEG with PROCOPT=K.
- A SENFLD statement cannot not appear within a SENSEG with PROCOPT=I or L, if the SENSEG refers to a logical child segment.
- If SENFLD statements are used within a SENSEG with PROCOPT=I or L, a SENFLD statement must be included for the segment sequence field, if it exists.
- This statement is not supported for MSDB and DEDB.

The format of the SENFLD statement is as follows:



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

START=
Specifies the starting position of this field relative to the beginning of the segment within the user's I/O area. *startpos* for the first byte of a segment is 1. *startpos* must be a decimal number whose value does not exceed 32 767.

REPLACE= or REPL=
Specifies whether or not this field can be altered on a replace call. You can specify NO or N. If omitted, REPLACE=YES (or Y) is the default.

REMARKS=
Optional user comments. A 1- to 256-character field.

If your comments contain special characters, such as commas or blank spaces, enclose the full comment string in single quotation marks.

The value specified on the REMARKS keyword cannot contain the following characters:

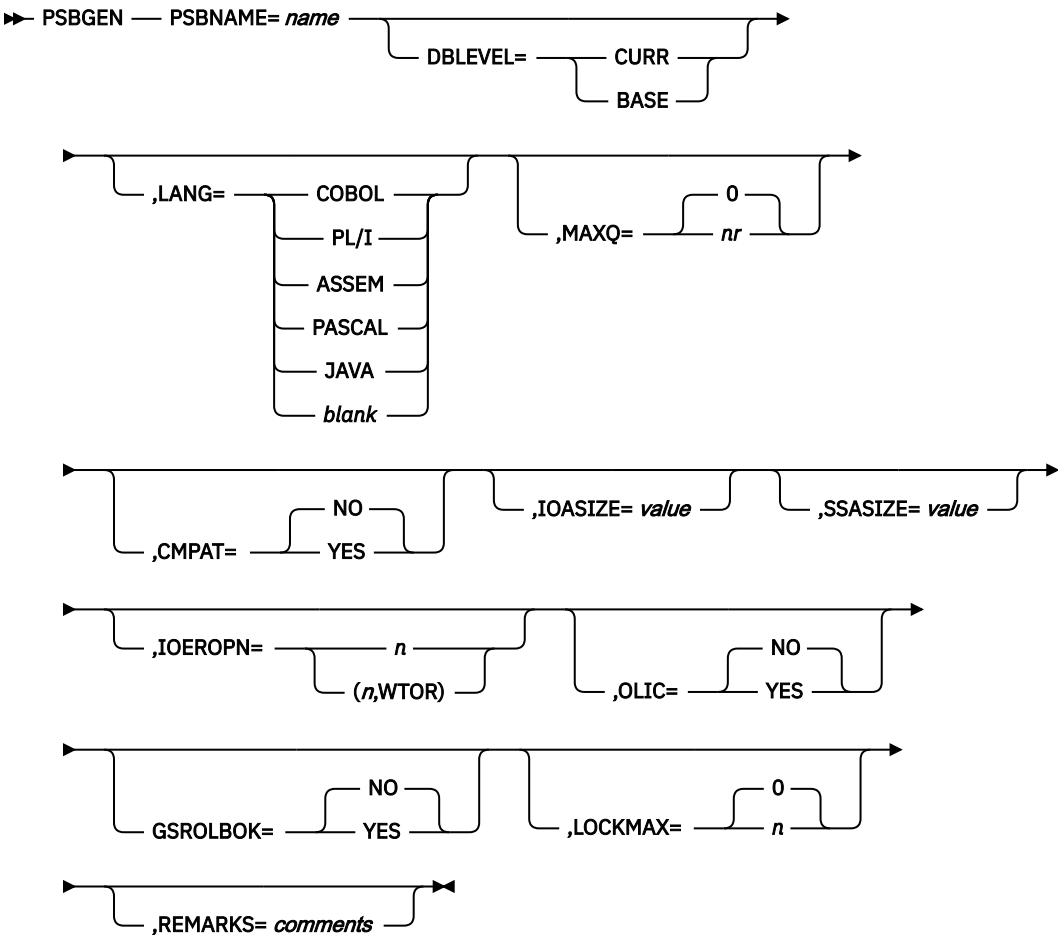
- Single quotation marks, except when they are used to enclose the full comment string. If a single quotation mark is entered before the end of the full comment string, the remainder of the comment string is truncated. The following examples show correct and incorrect usages of single quotation marks on the REMARKS keyword:

  CORRECT
  ```
  REMARKS='These remarks apply to the XYZ application'
  ```
  INCORRECT
  ```
  REMARKS='These remarks apply to the 'XYZ' application'
  ```
- Double quotation marks.
- Less than (<) symbols.
- Greater than (>) symbols.
- Ampersands (&).

PSBGEN statement

The PSBGEN statement specifies characteristics of the application program.

The following syntax diagram shows the format for the PSBGEN statement.



PSBNAME=

Specifies the parameter for the alphanumeric name of this PSB. The PSBNAME name must be an alphanumeric, 8-byte character string that follows standard naming conventions. This name becomes the load module name for the PSB in the library IMS.PSBLIB. If the program is to run in a message processing region, this name must be the same as the program load module name in the program library called IMS.PGMLIB No special characters can be used in the name.

Do not give a DBD the same name as an existing PSB. Using an existing name can cause unpredictable results: an error will occur at ACB generation time.

DBLEVEL=

When database versioning is enabled, specifies the version of the DBD that is used to return data to application programs that do not request a specific database version. For all application programs that use this PSB, the value specified here overrides the system default for DBLEVEL that is specified in the DFSDF*xxx* member of the IMS.PROCLIB data set.

The valid values for DBLEVEL are:

CURR

When DBLEVEL=CURR and an application program does not specify a version number on a call to access the database, IMS returns data to the application program by using the database structure that is defined by the active members in the ACB library. Usually, the active ACB members define the actual current structure of the physical database.

BASE

When DBLEVEL=BASE and an application program does not specify a database version number, IMS returns data to the application program by using the database structure that is defined by the DBD record in the IMS catalog that contains DBVER=0. DBVER=0 is the default version of a

DBD before a DBVER value is specified explicitly on a subsequent version of a DBD. If multiple instances of a version-0 DBD exist in the IMS catalog, IMS uses the instance that has the most recent timestamp.

Individual application programs can request DBD version numbers other than the one that is indicated by the DBLEVEL value by either specifying the DBD version number on the DBVER keyword of the PCB statement or on an INIT VERSION call.

When database versioning is disabled, the DBLEVEL parameter is ignored.

LANG=
An optional keyword that indicates the compiler language in which the message processing or batch processing program is written. If you specify OLICYES, LANGPLI is invalid. If your application program is written in C language, specify LANGASSEM.

CICS and the Language Environment for z/OS do not support LANGPASCAL.

You must specify LANGJAVA if the application is using the Java class libraries for IMS in a JMP region.

If you are using IMS PL/I applications that run in a compatibility mode using the PLICALLA entry point, you must specify LANGPLI.

MAXQ=
Is the maximum number of database calls with Qx command codes that can be issued between synchronization points. The maximum number is 32,767. The default value is zero.

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

IOASIZE=
Specifies the size (in bytes) of the largest I/O area used by the application program. The size specification is used to determine the amount of main storage reserved in the PSB pool to hold the control region copy of the user I/O area data during scheduling of this application program. If you do not specify this value, the IMS calculates a maximum default I/O area size. The default size is the total length of all of the sensitive segments in the longest possible path call. (The total length of the segment must be used, even if the application program is not sensitive to all fields in a segment.) The value specified is in bytes, with a maximum of 256000. However, the combined length of all concatenated segments to be returned to the application on a single path call must not exceed 65535 bytes.

If the PSB contains any field-sensitive segments, and IOASIZE is specified, the specified value is used only if it is larger than the IOASIZE calculated by the ACBGEN utility. The value of the IOASIZE that will be used is indicated. The major components of this pool requirement are IOASIZE and SSASIZE.

If STAT calls or the test program (DFSDDLT0) are used with this PSB, IOASIZE must be greater than 600 bytes.

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

If extended checkpoint/restart is used, IOASIZE must be set to a value equal to or greater than the larger of the following:

- I/O area needed to receive data from a GU call issued during restart, while repositioning DL/I databases that have a previous checkpoint (if this PSB contains any).
- Largest LRECL used in a GSAM data set that has a previous checkpoint.

Either the value pointed to by the third parameter (I/O AREA LEN) of the XRST CALL or the value of this parameter will be used, depending on which value is larger.

SSASIZE=
Specifies the maximum total length of all SSAs used by the application program. IMS uses the size specification to determine the amount of main storage reserved in the PSB work pool to hold a copy of the user's SSA strings during execution of this application program. If you do not specify this value,

the ACB utility program calculates a maximum SSA size to be used as a default. The size calculated is the maximum number of levels in any PCB within this PSB multiplied by 280. The value specified is in bytes, with a maximum of 256000.

Restriction: When you run IMS under CICS without DBCTL, the PSB work pool requirement cannot exceed 64KB.

The major components of this pool requirement are IOASIZE and SSASIZE. When the PSB is built into ACBLIB, ACB generation message DFS0589I indicates the PSB's total work pool space requirement.

Important: For Fast Path secondary index calls, an SSASIZE workarea holds the converted SSAs that accommodate the additional storage from SUBSEQ fields and number of qualifications. When a DL/I call is initiated, the converted SSAs are passed into the full-function database.

The default SSASIZE is specified as the default SSA size defined during ACBGEN plus 840 bytes.

If you specify an SSASIZE or if you use the default and the SSASIZE is not large enough, an AU status code is issued. To correct this problem, specify a larger SSASIZE in the PSB and rerun PSBGEN and ACBGEN to resolve the problem.

IOEROPN n

Is applicable only in batch-type regions (DLI or DBB). This parameter is not valid for CICS. The n subparameter is the condition code returned to the operating system when IMS terminates normally and one or more input or output errors occurred on any database during the application program execution. The n subparameter is a number from 0 to 4095.

If n=451, IMS terminates with a U451 abend instead of passing a condition code to the operating system. If n=451 and the IMS or the application program abends with an abend other than U451, and an I/O error has also occurred, a write-to-programmer of message DFS0426I is issued. This message indicates that an I/O error has occurred during execution and that a U451 abend has occurred if the actual abend has not.

If n=451, IMS terminates with abend U0451, even if the operator responds `CONT` to the DFS0451A message.

By using the IOEROPN parameter, you can set a unique JCL condition code when an I/O error occurs and test the condition code in subsequent job steps. If you do not specify this parameter, the return code passed from the application program is passed to the operating system and status codes and console messages are the only indications of database I/O errors.

OLIC=

Indicates whether the user of this PSB is authorized to execute the Online Database Image Copy utility or the Surveyor utility feature that runs as a BMP against a database named in this PSB. YES allows the Online Image Copy and the Surveyor utility feature; NO prohibits the Online Image Copy and the Surveyor utility feature. NO is the default. This parameter is invalid if any database PCB (a PCB with DB specified for the TYPE parameter) in the PSB specifies the L or LS processing options.

Exception: This parameter is not applicable to CICS, GSAM, HSAM, MSDB, or DEDB databases.

GSROLBOK=

Controls whether an internal ROLB call should be done to roll back non-GSAM database updates when:

- The application is a non-message-driven BMP.
- The PSB contains a PCB for a GSAM database.
- Db2 for z/OS reports a deadlock either on a thread create or on an SQL call.

YES means that the internal ROLB call is done and that the SQL code regarding the deadlock is returned to the application program. NO means that a user abend 777 occurs instead of the ROLB call. The default is NO.

LOCKMAX=

Indicates the maximum number of locks an application program can get at one time. *n* is a numeric value between 0 and 255. *n* is specified in units of 1000. For example, a specification of LOCKMAX=5 indicates a maximum of 5000 locks at one time.

The default value is 0. This indicates that there is no maximum number of locks that are allowed at one time.

If an application program runs for an extended time without committing, the locking done by IMS of database records and changes can accumulate. You can use the LOCKMAX parameter to prevent a single application program from consuming all locking storage and thereby causing other programs to abend.

You can override the LOCKMAX value specified on the PSBGEN statement at program execution by specifying LOCKMAX=0 (to turn off limit completely) or by specifying LOCKMAX=1 to 32767 on the dependent region (BMP, MPP, or IFP) or Batch (DBB or DLI). The value is in units of 1000. You can use this method to exceed the maximum value of 255 that can be specified on the PSBGEN statement LOCKMAX parameter.

REMARKS=

Optional user comments. A 1- to 256-character field.

If your comments contain special characters, such as commas or blank spaces, enclose the full comment string in single quotation marks.

The value specified on the REMARKS keyword cannot contain the following characters:

- Single quotation marks, except when they are used to enclose the full comment string. If a single quotation mark is entered before the end of the full comment string, the remainder of the comment string is truncated. The following examples show correct and incorrect usages of single quotation marks on the REMARKS keyword:

  CORRECT

  ```
  REMARKS='These remarks apply to the XYZ application'
  ```

  INCORRECT

  ```
  REMARKS='These remarks apply to the 'XYZ' application'
  ```

- Double quotation marks.
- Less than (<) symbols.
- Greater than (>) symbols.
- Ampersands (&).

There can be several PCB statements for message output and several PCB statements for databases, but only one PSBGEN in a PSB generation PCB statement deck. The PSBGEN statement must be the last statement in the deck preceding the END statement.

END statement

All PSB generation utility control statements must be followed by an END statement.

The END statement is required by the macro assembler to indicate the end of the assembly data.

Examples of the PSBGEN utility

These examples show how to use the PSBGEN utility to generate PSBs.

PSB generation examples

This following example shows a PSB generation for a message processing program to process the hierarchic data structure. The data structure contains segments: PARTMAST, CPWS, POLN, OPERTON, INVSTAT, and OPERSGMT.

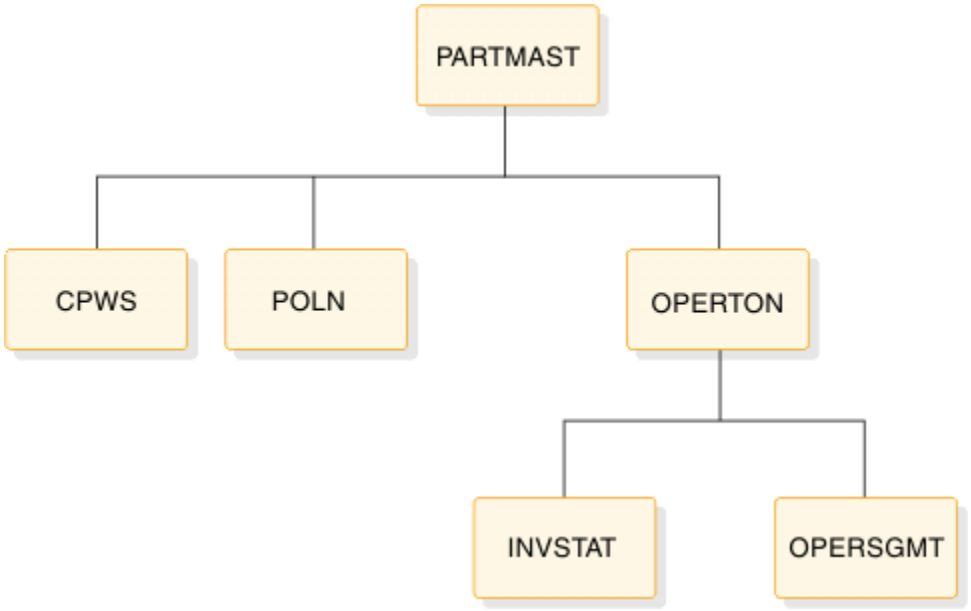


Figure 13. Sample hierarchic data structure

Example 1

This example shows output messages that are to be transmitted to logical terminals OUTPUT1 and OUTPUT2 as well as the terminal representing the source of input.

```
//PSBGEN   JOB
//        EXEC    PSBGEN,MBR=APPLPGM1
//C.SYSIN   DD   *

       PCB    TYPE=TP,NAME=OUTPUT1,PCBNAME=OUTPCB1
       PCB    TYPE=TP,NAME=OUTPUT2,PCBNAME=OUTPCB2
       PCB    TYPE=DB,DBDNAME=PARTMSTR,PROCOPT=A,KEYLEN=100
       SENSEG NAME=PARTMAST,PARENT=0,PROCOPT=A
       SENSEG NAME=CPWS,PARENT=PARTMAST,PROCOPT=A
       SENSEG NAME=POLN,PARENT=PARTMAST,PROCOPT=A
       SENSEG NAME=OPERTON,PARENT=PARTMAST,PROCOPT=A
       SENSEG NAME=INVSTAT,PARENT=OPERTON,PROCOPT=A
       SENSEG NAME=OPERSGMT,PARENT=OPERTON
       PSBGEN LANG=COBOL,PSBNAME=APPLPGM1
       END
/*
```

Example 2

This example shows these statements being used for a batch program, where programs using this PSB do not reference the telecommunications PCBs in the batch environment.

```
//PSBGEN   JOB
//        EXEC    PSBGEN,MBR=APPLPGM2
//C.SYSIN   DD   *

       PCB    TYPE=DB,DBDNAME=PARTMSTR,PROCOPT=A,KEYLEN=100
       SENSEG NAME=PARTMAST,PARENT=0,PROCOPT=A
       SENSEG NAME=CPWS,PARENT=PARTMAST,PROCOPT=A
       SENSEG NAME=POLN,PARENT=PARTMAST,PROCOPT=A
       SENSEG NAME=OPERTON,PARENT=PARTMAST,PROCOPT=A
       SENSEG NAME=INVSTAT,PARENT=OPERTON,PROCOPT=A
       SENSEG NAME=OPERSGMT,PARENT=OPERTON
       PSBGEN LANG=COBOL,PSBNAME=APPLPGM2
       END
/*
```

Example 3

This example shows that a PSB generation is being performed for a batch message processing program. The GSAM PCB is used by the application program to generate a report file.

```
//PSBGEN   JOB
//        EXEC    PSBGEN,MBR=APPLPGM3
//C.SYSIN   DD   *

       PCB    TYPE=TP,NAME=OUTPUT1
       PCB    TYPE=TP,NAME=OUTPUT2
       PCB    TYPE=DB,DBDNAME=PARTMSTR,PROCOPT=A,KEYLEN=100
       SENSEG NAME=PARTMAST,PARENT=0,PROCOPT=A
       SENSEG NAME=CPWS,PARENT=PARTMAST,PROCOPT=A
       PCB    TYPE=GSAM,DBDNAME=REPORT,PROCOPT=LS
       PSBGEN LANG=COBOL,PSBNAME=APPLPGM3
       END
/*
```

Example 4

This example shows that a PSB generation is being performed for a batch program. The PCB has been named (PRTMASTR). The PCB name is used on DLI calls that use the AIBTDLI interface.

```
//PSBGEN   JOB
//        EXEC    PSBGEN,MBR=APPLPGM4
//C.SYSIN   DD   *

 PCB    TYPE=DB,DBDNAME=PARTMSTR,PROCOPT=A,KEYLEN=100,PCBNAME=PARTMSTR
 SENSEG NAME=PARTMAST,PARENT=0,PROCOPT=A
 SENSEG NAME=CPWS,PARENT=PARTMAST,PROCOPT=A
 SENSEG NAME=POLN,PARENT=PARTMAST,PROCOPT=A
 SENSEG NAME=OPERTON,PARENT=PARTMAST,PROCOPT=A
 SENSEG NAME=INVSTAT,PARENT=OPERTON,PROCOPT=A
 SENSEG NAME=OPERSGMT,PARENT=OPERTON
 PSBGEN LANG=COBOL,PSBNAME=APPLPGM4
 END
/*
```

Example 5

This example shows that a PSB generation is being performed for a batch program. A label (PARTROOT) is being used to indicate the only root segment in the PCB. The PCB's address will be excluded from the PCB list that is passed to the application at entry.

```
//PSBGEN   JOB
//        EXEC    PSBGEN,MBR=APPLPGM5
//C.SYSIN   DD   *

 PARTROOT PCB TYPE=DB,DBDNAME=PARTMSTR,PROCOPT=A,LIST=NO
 SENSEG NAME=PARTMAST,PARENT=0,PROCOPT=A
 PSBGEN LANG=COBOL,PSBNAME=APPLPGM5
 END
/*
```

Field level sensitivity PSB generation example

The following figure shows a PCB for a batch program using field level sensitivity. The illustration shows the hierarchic order of the segments. The employee segment is at the first level. The office and employee project segments are at the second level. Outside of the hierarchic structure, but on the second level, the segment project is connected to the employee project segment.

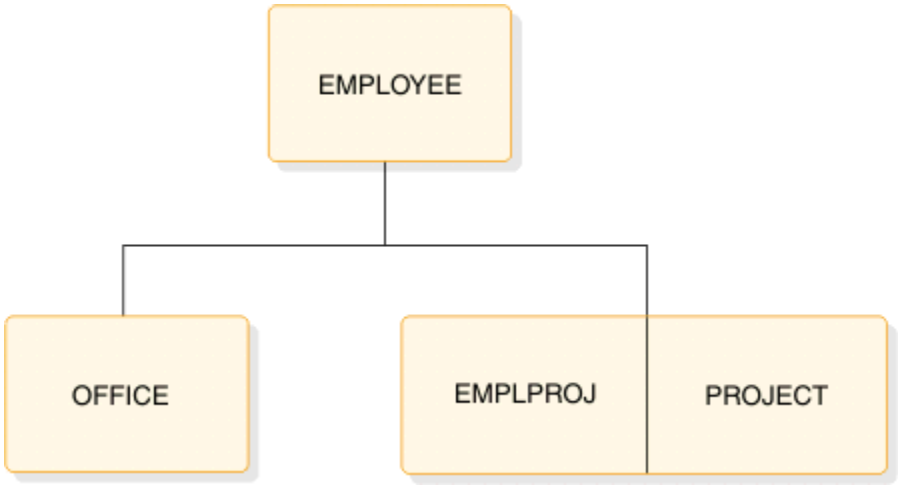


Figure 14. Sample field level sensitivity PSB generation

```
SEGMENT NAME      FIELD NAME      START LOCATION      LENGTH

  EMPLOYEE        EMPSSN               1               9
                  EMPLNAME            10              10
                  EMPFNAME            20               9
                  EMPMI               29               1
                  EMPADDR             30              30

OFFICE            OFNUMBER             1               5
                  OFPHONE              6               7
EMPLPROJ          EPFUNCTN             1              20
                  EPTIMEST            21               5
                  EPTIMCUR            26               5

PROJECT           PROJNUM              1               8
                  PROJTTLE             9              20
                  PROJSTRT            29               8
                  PROJEND             37               8
                  PROJSTAT            45               1


 //PSBGEN   JOB
 //        EXEC    PSBGEN,MBR=APPLPGM1
 //C.SYSIN   DD   *

        PCB      TYPE=DB,NAME=FISDBD1,PROCOPT=GRP,KEYLEN=20
        SENSEG   NAME=EMPLOYEE,PARENT=0
        SENFLD   NAME=EMPLNAME,START=13,REPL=NO
        SENFLD   NAME=EMPFNAME,START=1,REPL=NO
        SENFLD   NAME=EMPMI,START=11
        SENSEG   NAME=OFFICE,PARENT=EMPLOYEE
        SENSEG   NAME=EMPLPROJ,PARENT=EMPLOYEE
        SENFLD   NAME=PROJNUM,START=1
        SENFLD   NAME=PROJTITLE,START=10
        SENFLD   NAME=EPFUNCTN,START=35
        SENFLD   NAME=EPTIMEST,START=60
        SENFLD   NAME=EPTIMCUR,START =70
        PSBGEN   LANG=ASSEM,PSBNAME=APPLPGM1
        END
 /*
```

Fast Path PSB generation examples

The following two examples show sample Fast Path PSB Generations.

Example 1

This example shows the statements for an MSDB PSB containing eight PCBs.

```
 //PSBGEN   JOB
 //        EXEC    PSBGEN,MBR=APPLPGM1
```

```
 //C.SYSIN   DD   *

PCB    TYPE=DB,DBDNAME=MSDBLM01,PROCOPT=R,     NONTERMINAL-RELATED     X
       KEYLEN=4                                END OF PCB STATEMENT
SENSEG NAME=LDM,PARENT=0                       (DEFAULT)
PCB    TYPE=DB,DBDNAME=MSDBLM02,PROCOPT=R,     NONTERMINAL-RELATED     X
       KEYLEN=1
SENSEG NAME=LDM,PARENT=0
PCB    TYPE=DB,DBDNAME=MSDBLM03,PROCOPT=R,     NONTERMINAL-RELATED     X
       KEYLEN=2
SENSEG NAME=LDM,PARENT=0
PCB    TYPE=DB,DBDNAME=MSDBLM04,PROCOPT=R,     NONTERMINAL-RELATED     X
       KEYLEN=8                                TERM KEYS
SENSEG NAME=LDM,PARENT=0
PCB    TYPE=DB,DBDNAME=MSDBLM05,PROCOPT=R,     FIXED RELATED           X
       KEYLEN=8
SENSEG NAME=LDM,PARENT=0
PCB    TYPE=DB,DBDNAME=MSDBLM06,PROCOPT=A,     DYNAMIC RELATED         X
       KEYLEN=8
SENSEG NAME=LDM,PARENT=0
PCB    TYPE=DB,DBDNAME=MSDBLM06,PROCOPT=R,     DYNAMIC RELATED         X
       KEYLEN=8
SENSEG NAME=LDM,PARENT=0
PCB    TYPE=DB,DBDNAME=MSDBLM06,PROCOPT=G,     DYNAMIC RELATED         X
       KEYLEN=8
SENSEG NAME=LDM,PARENT=0
PSBGEN LANG=ASSEM,PSBNAME=APPLPGM1              END OF PSBGEN MACRO
END    END OF PSB GEN
 /*
```

Example 2

This example shows the statements for DEDB subset pointers.

```
 //PSBGEN   JOB
 //        EXEC    PSBGEN,MBR=APPLPGM1
 //C.SYSIN   DD   *

PCB    TYPE=DB,DBDNAME=MSDBLM01,PROCOPT=R,     NONTERMINAL-RELATED     X
PCB TYPE=DB,DBDNAME=X,PROCOPT=A,KEYLEN=100
SENSEG NAME=A,PARENT=C
SENSEG NAME=B,PARENT=A,SSPTR=((1,R),(2,U),(5))
SENSEG NAME=C,PARENT=B
SENSEG NAME=D,PARENT=A,SSPTR=((2,R))
PSBGEN LANG=COBOL,PSBNAME=APPLPGM1
END
 /*
```

Note:

1. SSPTR=((n,r))

   n
   Subset pointer number in this SENSEG

   r
   Sensitivity for the pointer (R: read, U: update)
2. If n and r are not specified, the pointer has no sensitivity.
3. If n is specified but r is not specified, the default is R (read sensitive).

Additional PSB generation examples

Example 1

The following figure shows a PSB generation that is being performed for a batch program. The illustration shows the hierarchic order of the segments. The Skill segment is at the first level. The Name segment (which is divided into payroll and skill) is at the second level. Address, Payroll, Expr, and Educ are on the third level.

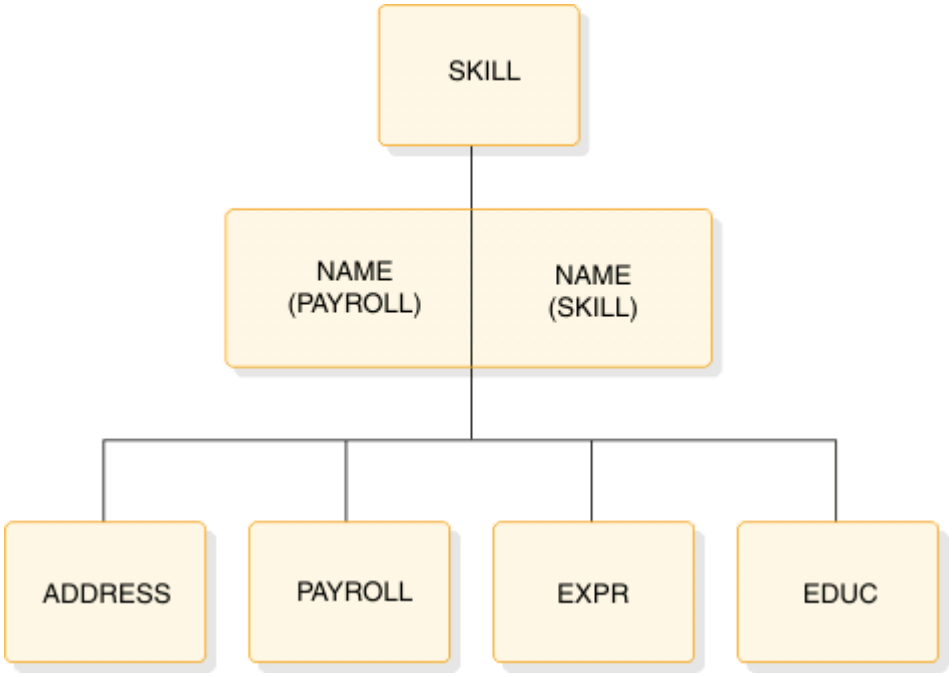


Figure 15. A PSBGEN statement used to define a DL/I database statement (example 1)

```
 //PSBGEN   JOB
 //        EXEC    PSBGEN,MBR=APPLPGM1
 //C.SYSIN   DD   *

PCB         TYPE=DB,DBDNAME=LOGIC1;PROCOPT=G,KEYLEN=151,POS=M
SENSEG      NAME=SKILL,PARENT=0,PROCOPT=A
SENSEG      NAME=NAME,PARENT=SKILL,PROCOPT=A
SENSEG      NAME=ADDRESS,PARENT=NAME,PROCOPT=A
SENSEG      NAME=PAYROLL,PARENT=NAME,PROCOPT=A
SENSEG      NAME=EXPR,PARENT=NAME,PROCOPT=A
SENSEG      NAME=EDUC,PARENT=NAME,PROCOPT=A
PSBGEN      LANG=COBOL,PSBNAME=APPLPGM1
END
 /*
```

Example 2:

The following figure shows a PSB generation that is being performed for a batch program. The illustration shows the hierarchic order of the segments. The NAME segment is at the first level. The NAMESK, ADDRESS, and PAYROLL segments are at the second level. The Expr and Educ segments are on the third level, connected to the NAMESK segment. Although the illustration separates the NAMESK segment into NAMESKIL and SKILL, the SENSEG statements do not define these as separate segments.

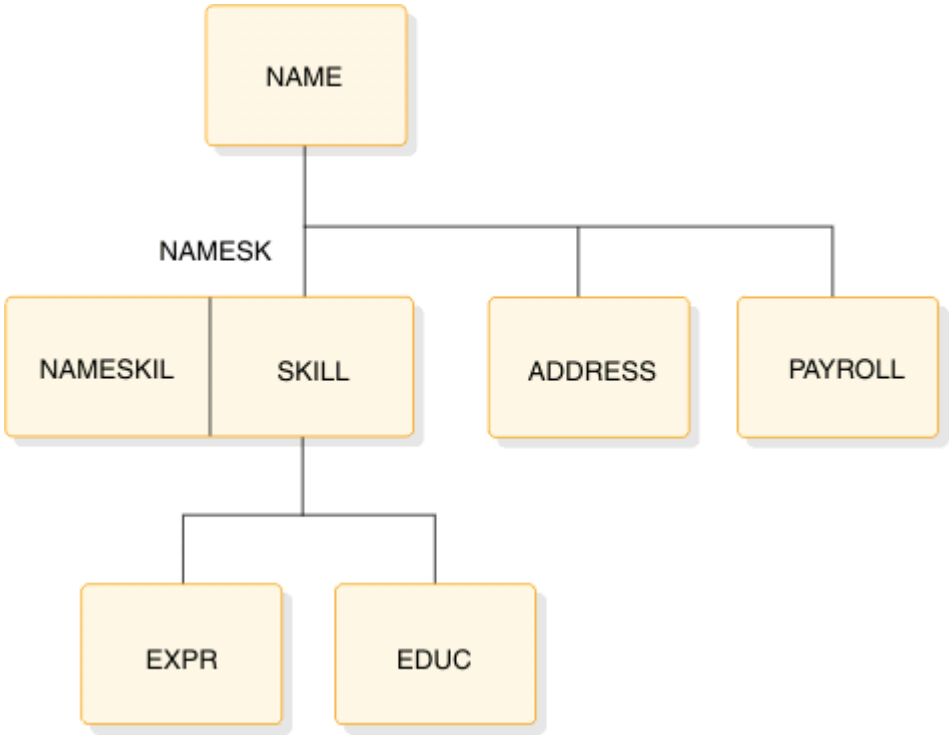


Figure 16. A PSBGEN PCB statement used to define a DL/I database PCB statement (example 2)

```
 //PSBGEN   JOB
 //        EXEC    PSBGEN,MBR=APPLPGM1
 //C.SYSIN   DD   *

PCB         TYPE=DB,DBDNAME=LOGICDB,PROCOPT=A,KEYLEN=241,POS=M
SENSEG      NAME=NAME,PARENT=0,PROCOPT=G
SENSEG      NAME=NAMESK,PARENT=NAME,PROCOPT=G
SENSEG      NAME=EXPR,PARENT=NAMESK,PROCOPT=G
SENSEG      NAME=EDUC,PARENT=NAMESK,PROCOPT=G
SENSEG      NAME=ADDRESS,PARENT=NAME,PROCOPT=G
SENSEG      NAME=PAYROLL,PARENT=NAME,PROCOPT=G
PSBGEN      LANG=PL/I,PSBNAME=APPLPGM1
END
 /*
```

Example 3:

The following figure shows a PSB that defines a logical relationship between segments in a DL/I database. The illustration shows the hierarchic order of the segments PARTMAST (the parent segment), CPWS, POLN, INVSTAT, and OPERSGMT (which are all first-level child segments of PARTMAST). The alternate statement sends output to logical terminal "OUTPUT". The PSBGEN statement saves this JCL as APPLPGM1 in the IMS.PSBLIB library.

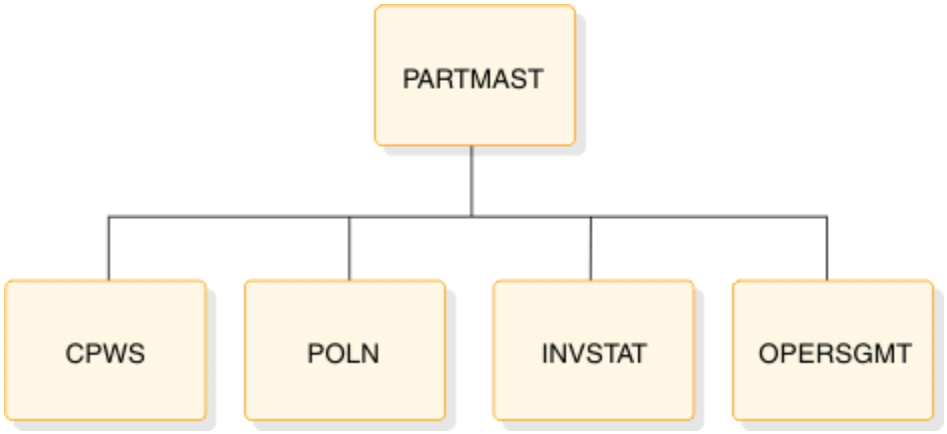


Figure 17. A PSBGEN PCB statement used to define a DL/I database PCB statement (example 3)

```
 //PSBGEN   JOB
 //        EXEC    PSBGEN,MBR=APPLPGM1
 //C.SYSIN   DD   *

PCB         TYPE=TP,LTERM=OUTPUT
PCB         TYPE=DB,DBDNAME=PARTMSTR,PROCOPT=GIDR,KEYLEN=100
SENSEG      NAME=PARTMAST,PARENT=0,PROCOPT=A
SENSEG      NAME=CPWS,PARENT=PARTMAST,PROCOPT=A
SENSEG      NAME=POLN,PARENT=PARTMAST,PROCOPT=A
SENSEG      NAME=INVSTAT,PARENT=PARTMAST,PROCOPT=A
SENSEG      NAME=OPERSGMT,PARENT=PARTMAST
PSBGEN      LANG=COBOL,PSBNAME=APPLPGM1
END
 /*
```

Example 4:

The following figure shows the JCL used to define the relationship between the POMSTR and POLNITEM segments from the DL/I database PODB. The alternate statements send output to applications with the transaction-code name "out1" and "out2".

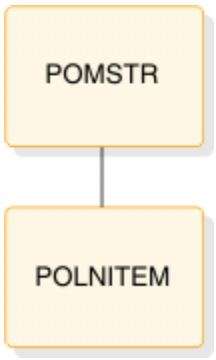


Figure 18. A PSBGEN PCB statement used to define a logical relationship and produce output

```
 //PSBGEN   JOB
 //        EXEC    PSBGEN,MBR=APPLPGM1
 //C.SYSIN   DD   *

PCB         TYPE=TP,NAME=OUT1
PCB         TYPE=TP,NAME=OUT2
PCB         TYPE=DB,DBDNAME=PODB,PROCOPT=GID,KEYLEN=200
SENSEG      NAME=POMSTR
SENSEG      NAME=POLNITEM,PARENT=POMSTR
PSBGEN      LANG=COBOL,PSBNAME=APPLPGM1
END
 /*
```

Examples of a sample problem with an application database

Examples five through ten use DBDNAME=DI21PART as a basis for the logical databases created with each example's JCL. The database contains segments PARTROOT, STANINFO, STOKSTAT, CYCCOUNT, and BACKORDR. PARTROOT is the parent segment. STANINFO and STOKSTAT are child segments of PARTROOT. CYCCOUNT and BACKORDR are child segments of STOKSTAT.

Example 1:

The following figure shows either a message switching or conversational message program. The JCL is saved as load module DFSSAM01 in the IMS.PSBLIB library.

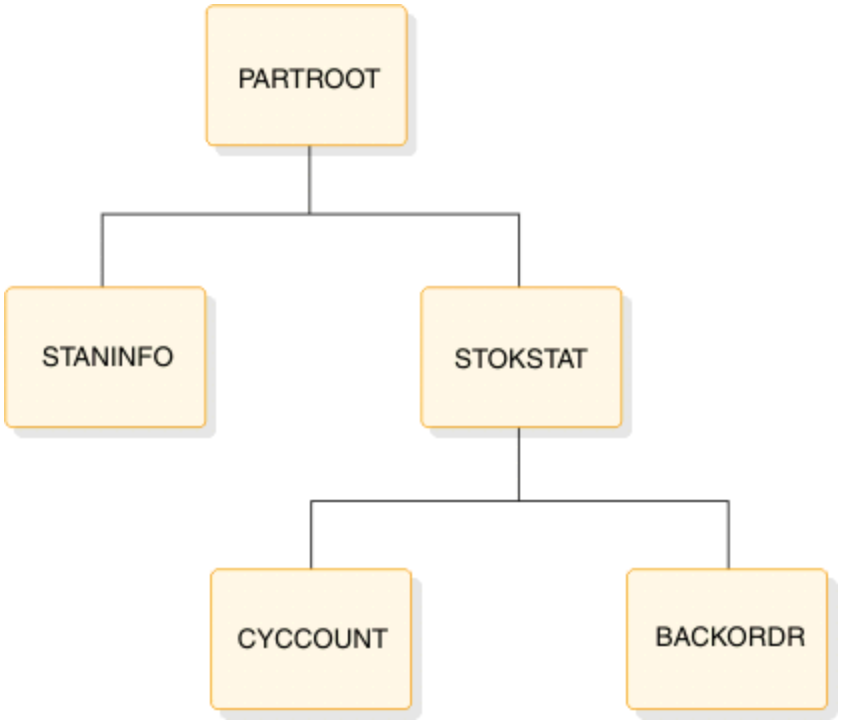


Figure 19. The data structure and JCL for a message switching or conversational message program

```
 //PSBGEN   JOB
 //        EXEC    PSBGEN,MBR=APPLPGM1
 //C.SYSIN   DD   *

SENSEG      NAME=BACKORDR,PARENT=STOKSTAT
PSBGEN      LANG=COBOL,PSBNAME=APPLPGM1
END
 /*
```

Example 2:

The JCL shown in the following figure defines a logical relationship between the PARTROOT and STANINFO segments (shown in the illustration with shading). The JCL is saved as load module DFSSAM02 in the IMS.PSBLIB library.

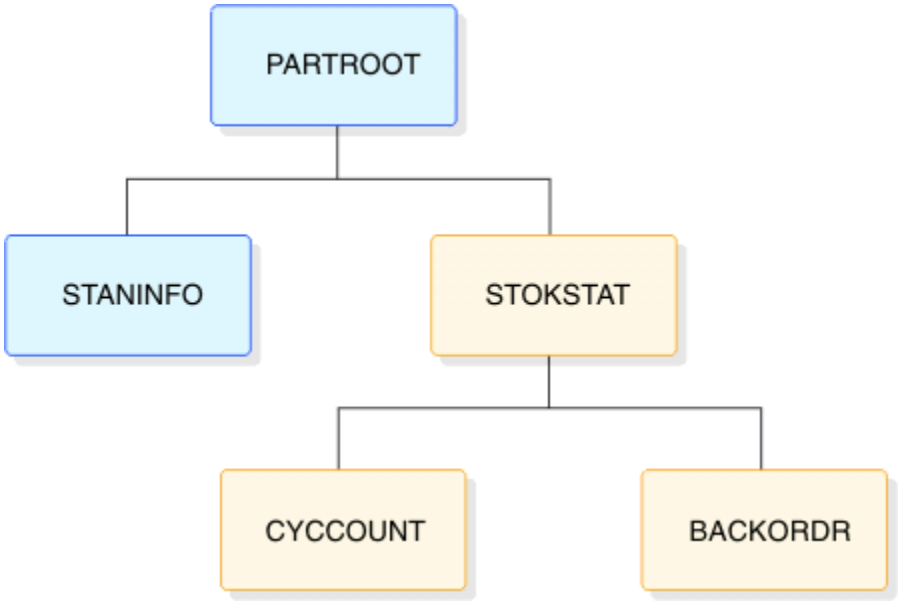


Figure 20. The data structure and JCL for a logical relationship in database DI21PART

```
 //PSBGEN   JOB
 //        EXEC    PSBGEN,MBR=APPLPGM1
 //C.SYSIN   DD   *

PCB         TYPE=DB,DBDNAME=DI21PART,PROCOPT=G,KEYLEN=19
SENSEG      NAME=PARTROOT
SENSEG      NAME=STANINFO,PARENT=PARTROOT
PSBGEN      LANG=COBOL,PSBNAME=APPLPGM1
END
 /*
```

Example 3:

The following figure defines the entire logical structure from the DL/I database DI21PART. The JCL is saved as load module DFSSAM03 in the IMS.PSBLIB library.

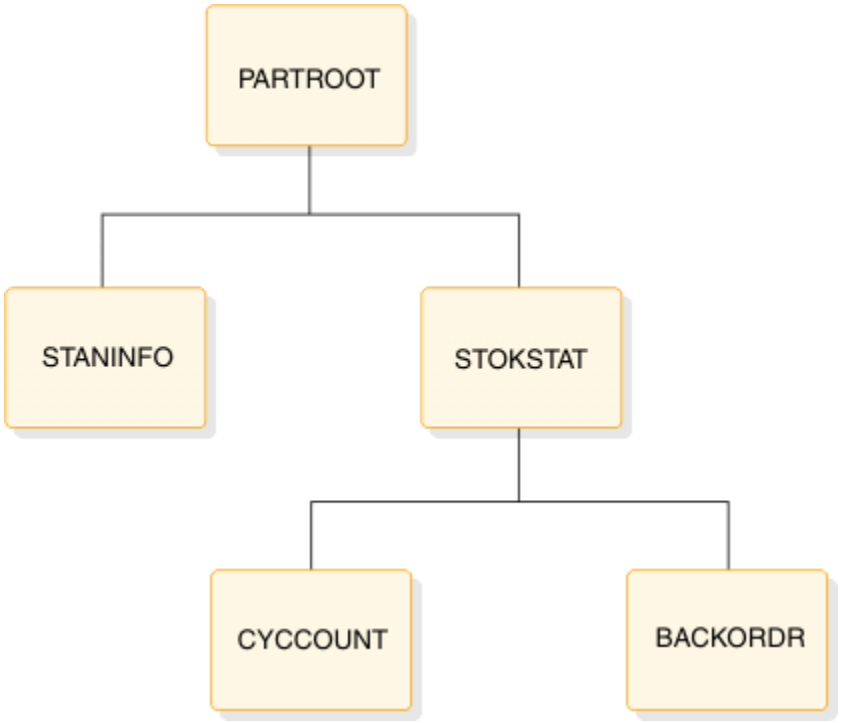


Figure 21. The data structure and JCL for a logical database defined from DL/I database DI21PART

```
 //PSBGEN   JOB
 //        EXEC    PSBGEN,MBR=APPLPGM1
 //C.SYSIN   DD   *

PCB         TYPE=DB,DBDNAME=DI21PART,PROCOPT=G,KEYLEN=43
SENSEG      NAME=PARTROOT
SENSEG      NAME=STANINFO,PARENT=PARTROOT
SENSEG      NAME=STOKSTAT,PARENT=PARTROOT
SENSEG      NAME=CYCCOUNT,PARENT=STOKSTAT
SENSEG      NAME=BACKORDR,PARENT=STOKSTAT
PSBGEN      LANG=COBOL,PSBNAME=APPLPGM1
END
 /*
```

Example 4:

The following figure defines the logical relationship between the PARTROOT and STOKSTAT segments (shown in the illustration with shading). The JCL also outputs to the logical terminal HOWARD and saves the JCL as load module DFSSAM03 in the IMS.PSBLIB library.

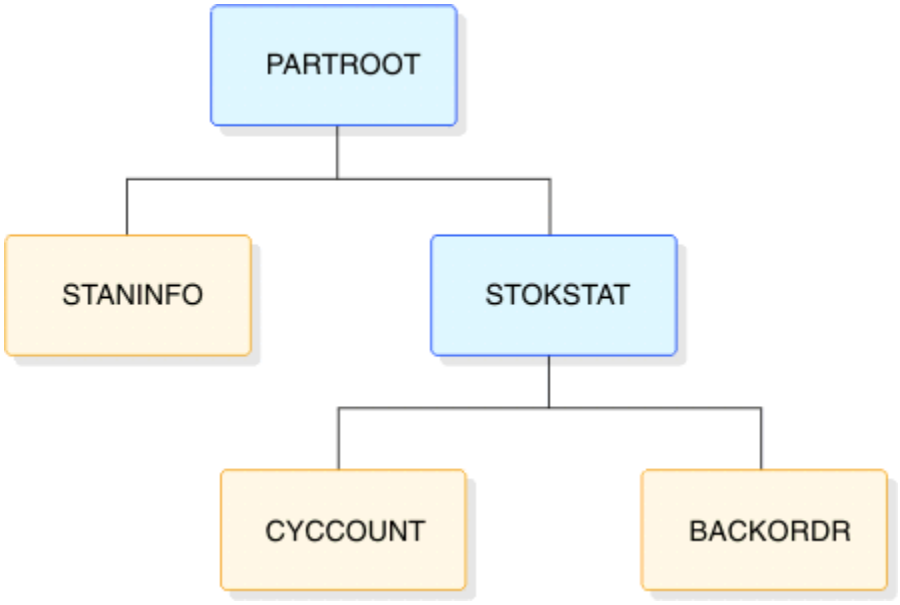


Figure 22. The data structure and JCL for a logical relationship in database DI21PART that produces output (part 1)

```
 //PSBGEN   JOB  MSGLEVEL=1
 //        EXEC    PSBGEN,MBR=APPLPGM1
 //C.SYSIN   DD   *

PCB         TYPE=TP,LTERM=HOWARD
PCB         TYPE=DB,DBDNAME=DI21PART,PROCOPT=A,KEYLEN=33
SENSEG      NAME=PARTROOT
SENSEG      NAME=STOKSTAT,PARENT=PARTROOT
PSBGEN      LANG=COBOL,PSBNAME=DFSSAM05
END
 /*
```

Example 5:

The following figure is identical to example 8, except this JCL is saved as load module DFSSAM06 in the IMS.PSBLIB library.

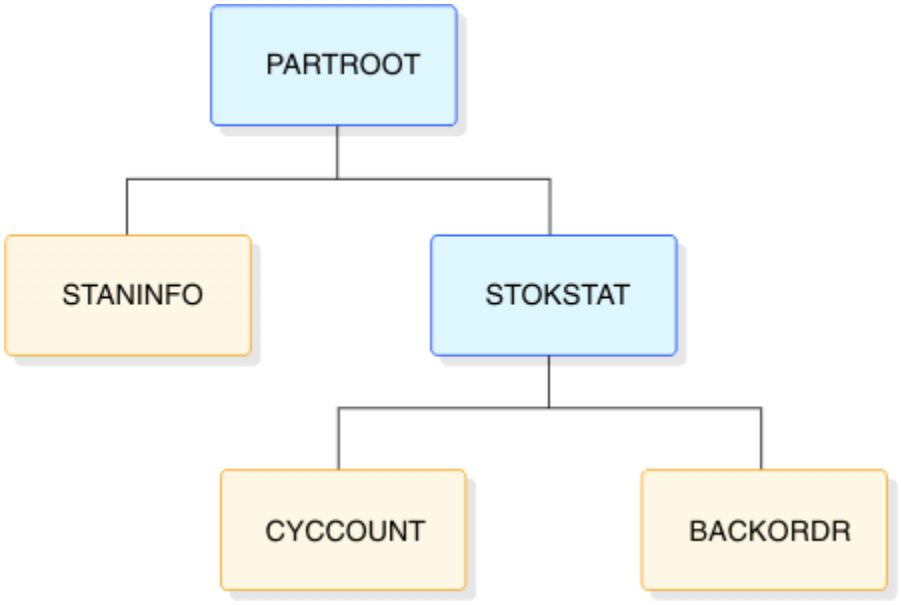


Figure 23. The data structure and JCL for a logical relationship in database DI21PART that produces output (part 2)

```
 //PSBGEN   JOB
 //        EXEC    PSBGEN,MBR=APPLPGM1
 //C.SYSIN   DD   *

PCB         TYPE=TP,LTERM=HOWARD
PCB         TYPE=DB,DBDNAME=DI21PART,PROCOPT=A,KEYLEN=33
SENSEG      NAME=PARTROOT
SENSEG      NAME=STOKSTAT,PARENT=PARTROOT
PSBGEN      LANG=COBOL,PSBNAME=APPLPGM1
END
 /*
```

Example 6:

The following figure defines the entire logical structure from the DL/I database DI21PART. The JCL is saved as load module DFSSAM07 in the IMS.PSBLIB library.

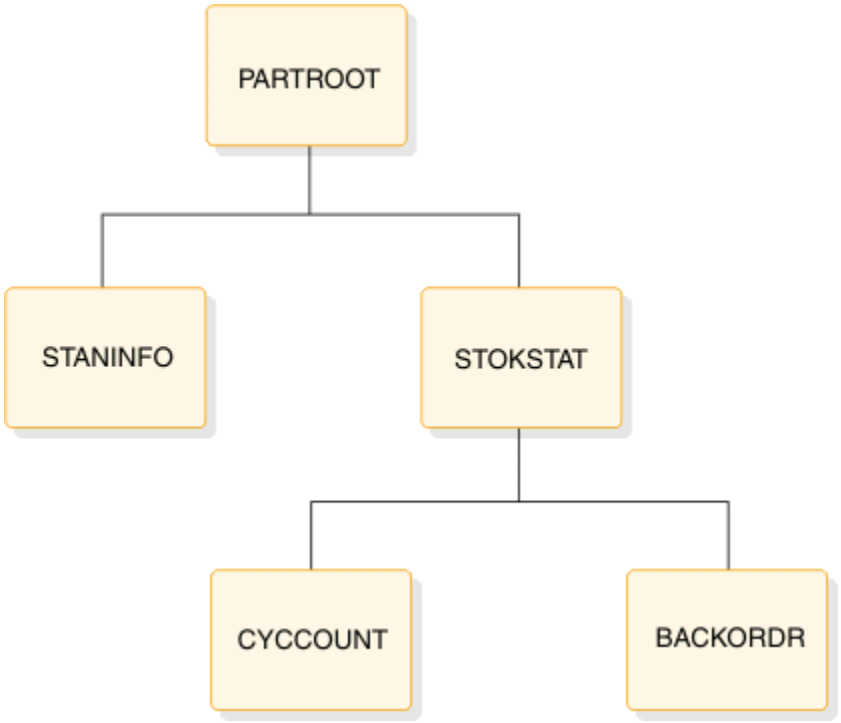


Figure 24. The data structure and JCL for a logical database defined from DL/I database DI21PART

```
 //PSBGEN   JOB
 //        EXEC    PSBGEN,MBR=APPLPGM1
 //C.SYSIN   DD   *

PCB         TYPE=DB,DBDNAME=DI21PART,PROCOPT=G,KEYLEN=43
SENSEG      NAME=PARTROOT
SENSEG      NAME=STANINFO,PARENT=PARTROOT
SENSEG      NAME=STOKSTAT,PARENT=PARTROOT
SENSEG      NAME=CYCCOUNT,PARENT=STOKSTAT
SENSEG      NAME=BACKORDR,PARENT=STOKSTAT
PSBGEN      LANG=COBOL,PSBNAME=APPLPGM1
END
 /*
```

Example of a shared secondary index

The following figure shows the database structure for this example. It shows a database, DTA3, that is indexed by three secondary indexes (X4, X5, and X6) in a shared secondary index database, X4. Each secondary index uses a different segment as both its index target segment and index source segment. Secondary index X4 uses DTA3 segment DA as its target/source segment. Secondary index X5 uses DTA3 segment DC as its target/source segment. Secondary index X6 uses DTA3 segment DE as its target/source segment.

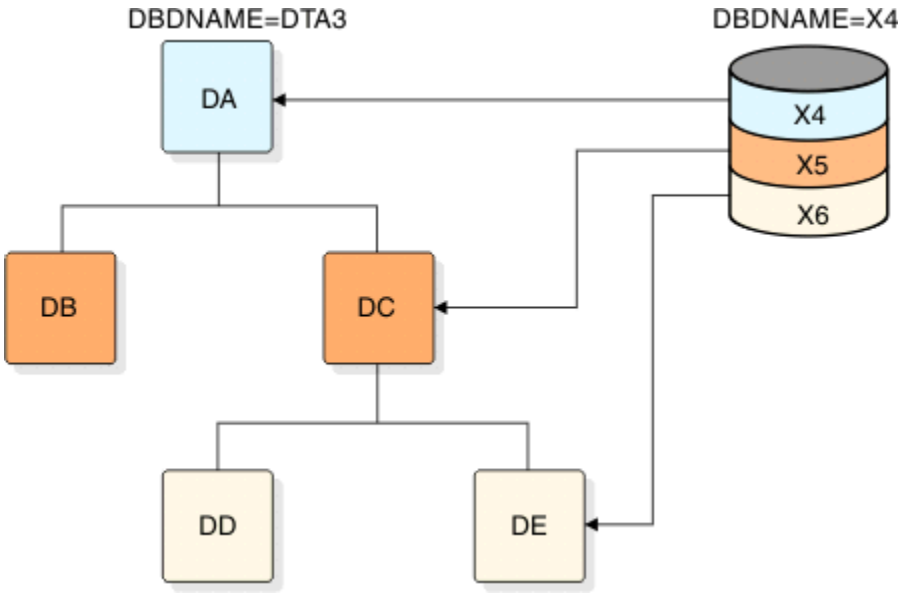


Figure 25. Database indexed by three secondary indexes in a shared secondary index database

The following figure shows the database structure for index through DA. It contains segments DA, DB, DC, DD, and DE.

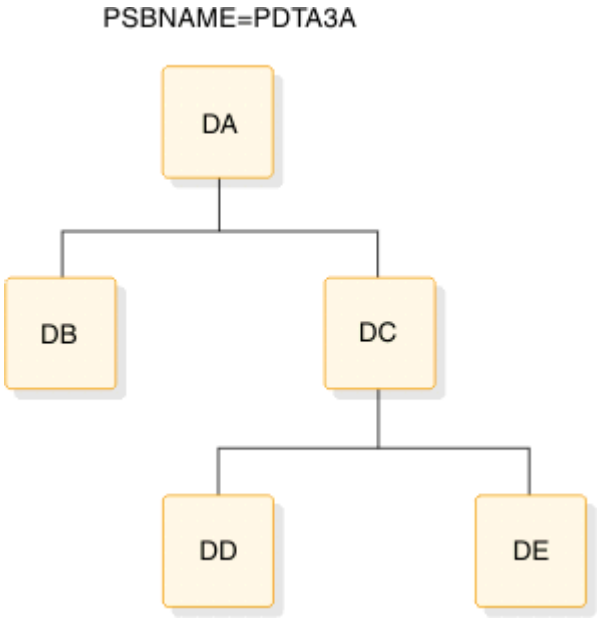


Figure 26. The data structure and JCL for index through segment DA

```
 //PSBGEN   JOB
 //        EXEC    PSBGEN,MBR=APPLPGM1
 //C.SYSIN   DD   *

PCB         TYPE=DB,DBDNAME=DTA3,PROCOPT=A,KEYLEN=15,PROCSEQ=X4
SENSEG      NAME=DA,PARENT=0
SENSEG      NAME=DB,PARENT=DA
SENSEG      NAME=DC,PARENT=DA,INDICES=X5
SENSEG      NAME=DD,PARENT=DC
SENSEG      NAME=DE,PARENT=DC,INDICES=X6
PSBGEN      LANG=COBOL,PSBNAME=APPLPGM1
END
 /*
```

The following figure shows the database structure for index through DC. It shows segment DC, DA, DD, and DE.

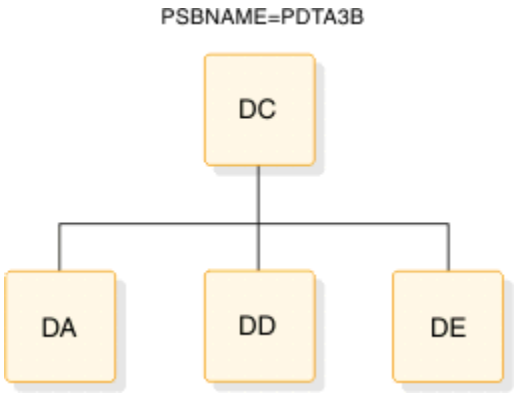


Figure 27. The data structure and JCL for index through segment DC

```
 //PSBGEN   JOB
 //        EXEC    PSBGEN,MBR=APPLPGM1
 //C.SYSIN   DD   *

PCB         TYPE=DB,DBDNAME=DTA3,PROCOPT=A,KEYLEN=15,PROCSEQ=X5
SENSEG      NAME=DC,PARENT=0
SENSEG      NAME=DA,PARENT=DC,INDICES=X4
SENSEG      NAME=DD,PARENT=DC
SENSEG      NAME=DE,PARENT=DC,INDICES=X6
PSBGEN      LANG=COBOL,PSBNAME=APPLPGM1
END
 /*
```

This database structure can also include, as a substructure, the database structure for index through DA.

The following figure shows the database structure for index through DE. It shows segments DE, DC, and DA.

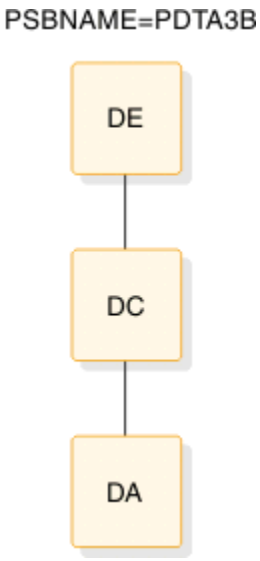


Figure 28. The data structure and JCL for index through segment DE

```
 //PSBGEN   JOB
 //        EXEC    PSBGEN,MBR=APPLPGM1
 //C.SYSIN   DD   *

PCB         TYPE=DB,DBDNAME=DTA3,PROCOPT=A,KEYLEN=15,PROCSEQ=X6
SENSEG      NAME=DE,PARENT=0
SENSEG      NAME=DC,PARENT=DE,INDICES=X5
SENSEG      NAME=DA,PARENT=DC,INDICES=X4
PSBGEN      LANG=COBOL,PSBNAME=APPLPGM1
END
 /*
```

This database structure can also include, as substructures, the database structures for indexes through DA and DC.

The PCB for INDEX database is shown as follows:

```
 //PSBGEN   JOB
 //        EXEC    PSBGEN,MBR=APPLPGM1
 //C.SYSIN   DD   *

PCB         TYPE=DB,DBDNAME=X4,PROCOPT=A,KEYLEN=5
SENSEG      NAME=X4A,PARENT=0
PCB         TYPE=DB,DBDNAME=X5,PROCOPT=A,KEYLEN=5
SENSEG      NAME=X5A,PARENT=0
PCB         TYPE=DB,DBDNAME=X6,PROCOPT=A,KEYLEN=5
SENSEG      NAME=X6A,PARENT=0
PSBGEN      LANG=COBOL,PSBNAME=APPLPGM1
END
 /*
```

Examples of Fast Path secondary indexes

The following example shows a Fast Path secondary index PCB in PSB using the PROCSEQD= parameter with the target segment is a root segment.

When PCB PCB2NDX is selected, the primary DEDB database, EDUCDB, is accessed using its Fast Path secondary index database, NAMESXDB. The PROCSEQD parameter specifies the name of the Fast Path secondary index database, NAMESXDB, to use to access the primary DEDB database, EDUCDB.

The target segment, COURSE segment, is a root segment. All segments under the root segment are accessible in the entire physical structure of the primary DEDB database using NAMESXDB secondary index.

```
PCB2NDX
  PCB      TYPE=DB,DBDNAME=EDUCDB,PROCOPT=A,KEYLEN=100,PROCSEQD=NAMESXDB
  SENSEG   NAME=COURSE,PARENT=0,PROCOPT=GR
  SENSEG   NAME=CLASS,PARENT=COURSE
  SENSEG   NAME=INSTRUCT,PARENT=CLASS        <<-- target segment
  SENSEG   NAME=STUDENT,PARENT=CLASS
  PSBGEN   PSBNAME=NAMEXPSB,LANG=COBOL
  END
```

The following example shows a Fast Path secondary index PCB in PSB using the PROCSEQD= parameter with the target segment is not a root segment.

When PCB PCB3NDX is selected, the primary DEDB database, EDUCDB, is accessed using its Fast Path secondary index database, INSTSXDB. The PROCSEQD parameter specifies the name of the Fast Path secondary index database, INSTSXDB, to use to access the primary DEDB database, EDUCDB.

The target segment, INSTRUCT segment, is not a root segment. Only the segments that are the direct parents of the target segment along the physical path from the root segment, and all child segments of the target segment are accessible in the physical structure of the primary DEDB database when the primary DEDB database is accessed using INSTSXDB secondary index.

Note: The STUDENT segment is not accessible because it is neither a direct parent segment nor a child segment of the target segment, INSTRUCT segment.

```
PCB3NDX
  PCB      TYPE=DB,DBDNAME=EDUCDB,PROCOPT=GR,KEYLEN=100,PROCSEQD=INSTSXDB
  SENSEG   NAME=COURSE,PARENT=0
  SENSEG   NAME=CLASS,PARENT=COURSE
  SENSEG   NAME=INSTRUCT,PARENT=CLASS        <<-- target segment
  PSBGEN   PSBNAME=NAMEXPSB,LANG=COBOL
  END
```

The following example shows a Fast Path secondary index PCB in PSB using the PROCSEQD= parameter with the target segment is not a root segment.

When PCB PCB4NDX is selected, the primary DEDB database, EDUCDB, is accessed using its Fast Path secondary index database, CLASSXDB. The PROCSEQD parameter specifies the name of the Fast Path secondary index database, CLASSXDB, to use to access the primary DEDB database, EDUCDB.

The target segment, CLASS segment, is not a root segment. Only the segments that are the direct parents of the target segment along the physical path from the root segment, and all child segments of the target segment are accessible in the physical structure of the primary DEDB database when the primary DEDB database is accessed using the CLASSXDB secondary index database. The STUDENT segment is accessible because it is a child segment of the target segment, CLASS segment.

```
PCB4NDX
  PCB      TYPE=DB,DBDNAME=EDUCDB,PROCOPT=GR,KEYLEN=100,PROCSEQD=CLASSXDB
  SENSEG   NAME=COURSE,PARENT=0
  SENSEG   NAME=CLASS,PARENT=COURSE
  SENSEG   NAME=INSTRUCT,PARENT=CLASS
  SENSEG   NAME=STUDENT,PARENT=CLASS
  PSBGEN   PSBNAME=NAMEXPSB,LANG=COBOL
  END
```

The following example shows a Fast Path secondary index as a database.

```
PCB2XDB
  PCB      TYPE=DB,DBDNAME=NAMESXDB,PROCOPT=GR,KEYLEN=10
  SENSEG   NAME=NAMEXSEG,PARENT=0
  PSBGEN   PSBNAME=DB2DXPSB,LANG=COBOL
  END
```

Running the PSBGEN procedure

The DFSPSBGN procedure is generated in the ADFSPROC and SDFSPROC data sets. The DFSPROCB JCL can then be used to rename DFSPSBGN to PSBGEN.

Recommendations

Currently, no recommendations are documented for the PSB Generation utility.

The first step, Step C, an operating system assembly, is performed after the procedure is invoked. The second step, Step L, is a bind which takes the assembly output from Step C and places the PSBs in IMS.PSBLIB.

Procedure statement

The following figure shows the procedure statement. The list following the figure defines the parameters used in the statement.

```
//       PROC MBR=TEMPNAME,SOUT=A,RGN=0M,SYS2=,
//               NODE1=IMS,
//               NODE2=IMS
//C      EXEC PGM=ASMA90,REGION=&RGN,
//            PARM=(OBJECT,NODECK,NODBCS,
//            'SIZE(MAX,ABOVE)')
//SYSLIB   DD DSN=&NODE2..&SYS2.SDFSMAC,DISP=SHR
//SYSLIN   DD UNIT=SYSDA,DISP=(,PASS),
//         SPACE=(80,(100,100),RLSE),
//         DCB=(BLKSIZE=80,RECFM=F,LRECL=80)
//SYSPRINT DD SYSOUT=&SOUT,DCB=BLKSIZE=1089,
//         SPACE=(121,(300,300),RLSE,,ROUND)
//SYSUT1   DD UNIT=SYSDA,DISP=(,DELETE),
//         SPACE=(CYL,(10,5))
//L      EXEC PGM=IEWL,PARM='XREF,LIST',
//            COND=(0,LT,C),REGION=4M
//SYSLIN   DD DSN=*.C.SYSLIN,DISP=(OLD,DELETE)
//SYSPRINT DD SYSOUT=&SOUT,DCB=BLKSIZE=1089,
//         SPACE=(121,(90,90),RLSE)
//SYSLMOD  DD DISP=SHR,
//         DSN=&NODE1..&SYS2.PSBLIB(&MBR)
```

```
//SYSUT1   DD UNIT=(SYSDA,SEP=(SYSLMOD,SYSLIN)),
//         SPACE=(1024,(100,10),RLSE),DISP=(,DELETE)
```

MBR=
Is the name of the PSB generated. This name should be the same as the name specified on the PSBNAME= parameter of the PSBGEN statement. If this precaution is not followed, a user ABEND 929 can occur during execution, or message DFS929I ("BLDL FAILED FOR MEMBER") can be received during an ACB generation "BUILD PSB" operation.

SOUT=
Specifies the SYSOUT class. The default is A.

RGN=
Specifies the region size for execution of the PSBGEN utility. The default is 512KB.

SYS2=
Specifies an optional second level dsname qualifier for those data sets which are designated as "Optional Replicate" in an XRF complex. When specified, the parameter must be enclosed in quotes and must include a trailing period, for example, SYS2='IMSA.'.

Step C

Step C is the assembly step.

DD statements

SYSIN DD
Defines the input data sets to step C. These DD statements must be provided when invoking the procedure.

Step L

Step L is the bind step.

Example: This step can be run using AMODE=31, RMODE=24 instead of the default AMODE=24, RMODE=24 by adding AMODE=31 to the bind EXEC statement PARM list as shown as follows.

```
//L      EXEC  PGM=IEWL,PARM='XREF,LIST,AMODE=31',
//             COND=(0,LT,C),REGION=120K
```

If you do not specify different values for AMODE or RMODE, the default values are in effect. You must always run the bind step with RMODE=24.

DD statements

SYSLMOD DD
Defines an output partitioned data set, IMS.PSBLIB, for the binder.

Invoking the procedure

The JCL statements in the following figure are used to invoke the PSBGEN procedure.

```
//PSBGEN   JOB
//        EXEC    PROC=PSBGEN,MBR=TEMPNAME
//C.SYSIN   DD   *

          PCB
       SENSEG     (The control statements for PSB generation)
       PSBGEN     PSBNAME=TEMPNAME
       END
/*
```

Part 2. IMS catalog utilities

Use the IMS catalog utilities to perform a variety of tasks for an IMS catalog.

Each topic introduces how the utility works, defines requirements and restrictions for its use, and provides examples.

© Copyright IBM Corp. 1974, 2024

Chapter 6. ACB Generation and Catalog Populate utility (DFS3UACB)

Use the ACB Generation and Catalog Populate utility (DFS3UACB) to generate ACB members in an IMS.ACBLIB data set, create the corresponding metadata records in the IMS catalog, and, if your IMS system manages ACBs, add the resulting ACBs as pending changes to the staging data set of the IMS catalog, all in a single job step.

In IMS systems that manage ACBs by using the catalog, the generation of ACBs into an ACB library is not required. However, you can still use the DFS3UACB utility to generate ACBs and populate the IMS catalog if you have tools or utilities that have not yet been updated for the IMS management of ACBs.

Populating the IMS catalog in the same job step as the generation of the ACB members ensures that the IMS catalog and your ACB library are consistent with each other.

The DFS3UACB utility calls the ACB Maintenance utility to generate ACB members and then, in the same job step, calls the IMS Catalog Populate utility (DFS3PU00) to populate the IMS catalog with records that correspond to the generated ACB members.

You must specify control statements for the ACB Maintenance utility by using the SYSIN DD statement. If you do not specify control statements for the ACB Maintenance utility, DFS3UACB terminates immediately.

You may provide execution parameters for the ACB Maintenance utility. To do so, provide them as execution parameters in the PARM parameter for the DFS3UACB utility. In these parameters do not specify the POSTCOMP option if the ACBCATWK DD statement is used.

You must specify the execution parameters for the DFS3PU00 utility, including the name of the DFSDFxxx PROCLIB member that supports your IMS catalog, by using the DFS3PPRM DD statement. The DFS3UACB utility passes the execution parameters to the DFS3PU00 utility at the start of the population phase.

If the IMS management of ACBs is enabled, you must specify the MANAGEDACBS= control statement on the SYSINP DD statement to make the generated ACBs available to the online IMS system. Depending on the specification, the DFS3PU00 utility either adds the ACBs to the staging data set of the IMS catalog for later activation or, if the IMS system is offline or you are setting up managed ACBs for the first time, adds them directly to the IMS directory data sets. IMS loads the ACBs from the directory data sets during startup.

The DFS3UACB utility can populate the catalog in load mode or in update mode. When load mode is used, any existing records in the IMS catalog are discarded.

The PSB that you specify in the utility JCL determines which access mode the utility uses to access the IMS catalog. You can specify the following PSBs in the DFS3UACB utility JCL:

- DFSCPL00, to perform an initial load of the IMS catalog
- DFSCP001, to insert records into an existing IMS catalog
- DFSCP000, to estimate the space requirements of the IMS catalog data sets

Subsections:

- “Restrictions” on page 314
- “Prerequisites” on page 314
- “Requirements” on page 314
- “Recommendations” on page 314
- “Input and output” on page 315
- “JCL specifications” on page 315
- “ACB generation control statements” on page 319

© Copyright IBM Corp. 1974, 2024

- “Catalog population control statements” on page 323
- “Return codes” on page 327

Restrictions

The DFS3UACB utility runs in a stand-alone region under z/OS control.

The DFS3UACB utility does not accept JCL parameters that are specified on the PARM= keyword of the EXEC= statement. To specify the execution parameters that are required to populate the IMS catalog, use the DFS3PPRM DD statement instead.

If you are running the DFS3UACB utility against more than one version of IMS in separate LPARs in an IMSplex environment, run the utility in separate jobs. For each job, use the SCHENV parameter to ensure that each job goes to the correct LPAR.

Prerequisites

If you are loading an IMS catalog for the first time, ensure that the IMS catalog is properly configured. The following steps must be complete before running the DFS3UACB utility to load an IMS catalog for the first time:

- The DBD and PSB load modules for the IMS catalog are in your DBD and PSB libraries.
- The ACB member for the IMS catalog was generated and is in the active ACB library.
- The IMS catalog HALDB master database and partitions are defined in either the RECON data set or, if the target IMS catalog is not supported by DBRC, in an IMS catalog partition definition data set.
- The CATALOG section of the DFSDFxxx PROCLIB member is properly coded.

To use the SHARE option to add DOPT PSBs to an online IMS catalog, extended sharing of PDSE data sets must be enabled in z/OS before the Populate utility first sets up the IMS management of ACBs. For more information, see z/OS: Specifying Extended PDSE Sharing in a Multiple-System Environment.

Requirements

IMS conforms to z/OS rules for data set authorization. If an IMS job step is authorized, all libraries used in that job step must be authorized. To run an IMS batch region as unauthorized, a non-authorized library must be concatenated to IMS.SDFSRESL.

You must specify control statements for the ACB Maintenance utility by using the SYSIN DD statement.

You must specify the execution parameters that the DFS3UACB utility passes to the DFS3PU00 utility by using the DFS3PPRM DD statement.

If the IMS management of ACBs is enabled, you must specify the MANAGEDACBS= control statement on the SYSINP DD statement to make the generated ACBs available to the online IMS system.

Recommendations

If you are updating an existing IMS catalog, consider creating an image copy of the IMS catalog data sets. If the IMS catalog is registered with DBRC, you can use the DBRC command GENJCL.IC to back up the catalog. If you defined the IMS catalog in an IMS Catalog partition definition data set, you must use standard image copy JCL.

The storage requirement is highly dependent on the total number of ACBLIB members because they have to be kept in storage. Consider increasing the job's region size to prevent a GETMAIN failure that will cause abend U1002 reason code 4 to be issued.

Input and output

The DFS3UACB utility uses the following input and output data sets.

Required data sets:

- IMS.ACBLIB data set. After creating the ACB members in the ACBLIB data set, the utility uses the ACBLIB data set as input when creating the records in the IMS catalog.
- DBDLIB data set.
- PSBLIB data set.
- ACBCATWK data set. The ACBCATWK data set is both an input and output data set. The ACB Maintenance utility uses it as an output data set to record the ACB members that are generated. The DFS3PU00 utility reads the ACBCATWK data set as input to improve performance when populating the IMS catalog.
- IMS.PROCLIB data set. The DFS3UACB utility reads the DFSDFxxx member in the IMS.PROCLIB data set.
- SYSIN control statements.
- An input data set or inline statement that contains execution parameters for the DFS3PU00 utility, which the DFS3UACB utility calls internally to populate the IMS catalog. This data set or inline statement is specified by using a DFS3PPRM DD statement.
- SYSPRINT messages.

Optional data sets:

- An input data set of COMPCTL IEBCOPY control statements.

The primary output of the DFS3UACB utility is the ACB library members and the records of the IMS catalog. The utility loads the ACB members into the inactive IMS.ACBLIB data set. The catalog records are stored in the IMS catalog data set (DFSCD000).

Optionally, the DFS3UACB utility can output a list of the generated ACB members to a data set referenced by the ACBCATWK DD statement. Maintaining a list of the generated ACB members greatly improves the performance of the population phase of the DFS3UACB utility.

The DFS3UACB utility also writes messages and statistical information to the SYSPRINT data set.

The DFS3UACB utility outputs to the SYSUT3 and SYSUT4 IEBCOPY utility data sets.

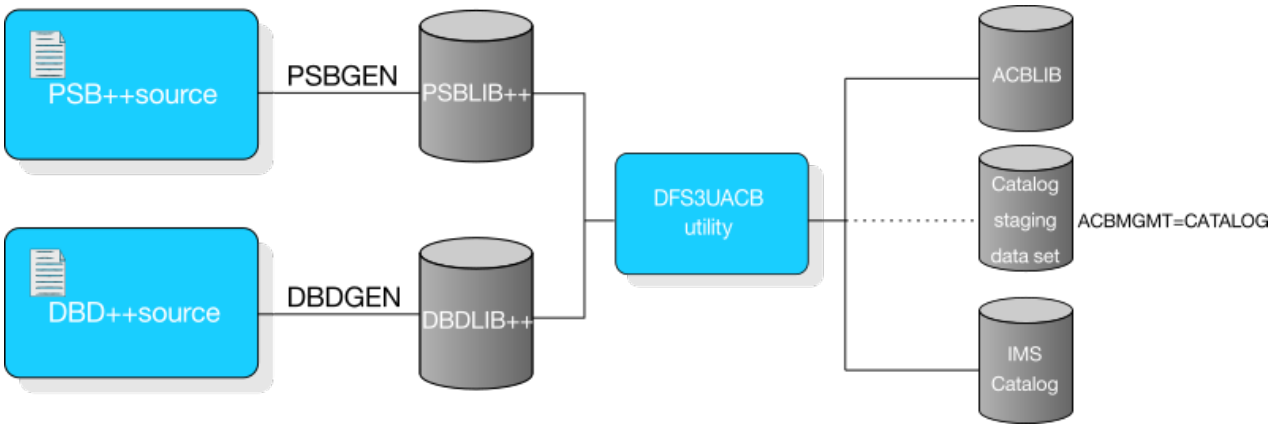


Figure 29. ACBGEN and Catalog Populate utility input and output

JCL specifications

Two examples of the DFS3UACB utility JCL are provided in the following subsections. One example shows the JCL for updating an existing catalog. The second example shows the JCL for loading an IMS catalog. Loading an IMS catalog deletes any existing records in the IMS catalog.

DFS3UACB utility JCL statements for update mode

The following is an example of the JCL statements that can be used to generate ACB members and catalog records for an existing IMS catalog by using the DFS3UACB utility. In the JCL, the PSB DFSCP001 is specified on the DFS3PPRM DD statement to access the IMS catalog in update mode.

```
//ACBPOPUP  JOB 'IMS SYSTEM',CLASS=K,MSGLEVEL=(1,1),REGION=0M
//*
//*******************************************************************
//* DFS3UACB GENERATES ACB MEMBERS IN AN ACB LIBRARY BY CALLING THE
//* ACB MAINTENANCE UTILITY. IN THE SAME JOB STEP,
//* DFS3UACB INSERTS RECORDS IN THE EXISTING IMS CATALOG BY CALLING
//* THE IMS CATALOG POPULATE UTILITY (DFS3PU00)
//*******************************************************************
//*
//ACBCATT EXEC PGM=DFS3UACB,REGION=0M
//*
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//SYSOUT   DD  SYSOUT=A
//SYSABEND DD  SYSOUT=*
//IMS      DD  DSN=IMS.PSBLIB,DISP=SHR
//         DD  DSN=IMS.DBDLIB,DISP=SHR
//ACBCATWK DD  SPACE=(CYL,(1,1)),UNIT=SYSDA
//*
//*******************************************************************
//* ACBGEN DATASETS
//*******************************************************************
//IMSACB   DD  DSN=IMS.ACBLIB,DISP=OLD
//SYSUT3   DD  UNIT=SYSDA,SPACE=(80,(100,100))
//SYSUT4   DD  UNIT=SYSDA,SPACE=(256,(100,100)),DCB=KEYLEN=30
//*******************************************************************
//* ACBGEN INPUT PARMS TO UPDATE ACBLIB
//*******************************************************************
//SYSIN    DD  *
        BUILD PSB=psbname (same as needed for ACBGEN)
/*
//*******************************************************************
//* POPULATE UTILITY DATASETS
//*******************************************************************
//IMSACB01 DD  DSN=*.IMSACB,DISP=OLD      DO NOT REPLACE ASTERISK
//SYSINP   DD  *              ISRTLIST DUPLIST /*
//DFSVSAMP DD  .......      BUFFER POOL DEFINITIONS
//IEFRDER  DD  .......      LOG DATASET FOR CATALOG UPDATES
//IEFRDER2 DD  .......      LOG DATASET FOR CATALOG UPDATES
//*******************************************************************
//* UPDATE INPUT PARMS FOR IMS CATALOG POPULATE UTILITY
//*******************************************************************
//DFS3PPRM DD  *
DLI,DFS3PU00,DFSCP001,,,,,,,,,,Y,Y,irlmidN,,,,,,,,,,DFSDF=CAT
/*
//
```

DFS3UACB utility JCL statements for load mode

The following is an example of the JCL statements that can be used to generate ACB members and load catalog records into an IMS catalog by using the DFS3UACB utility.

In the example, the PSB DFSCPL00 is specified on the DFS3PPRM DD statement to access the IMS catalog in load mode.

Attention: Running the DFS3UACB utility in load mode deletes any existing records in an IMS catalog.

```
//ACBPOPLD  JOB 'IMS SYSTEM',CLASS=K,MSGLEVEL=(1,1),REGION=0M
//*
//*******************************************************************
//* DFS3UACB GENERATES ACB MEMBERS IN AN ACB LIBRARY BY CALLING THE
//* ACB MAINTENANCE UTILITY. IN THE SAME JOB STEP,
//* DFS3UACB LOADS RECORDS IN THE IMS CATALOG BY CALLING
//* THE IMS CATALOG POPULATE UTILITY (DFS3PU00)
//*******************************************************************
//*
//ACBCATT EXEC PGM=DFS3UACB,REGION=0M
//*
```

```
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//SYSOUT   DD  SYSOUT=A
//SYSABEND DD  SYSOUT=*
//IMS      DD  DSN=IMS.PSBLIB,DISP=SHR
//         DD  DSN=IMS.DBDLIB,DISP=SHR
//ACBCATWK DD  SPACE=(CYL,(1,1)),UNIT=SYSDA
//*
//*****************************************************************
//* DATASETS USED ONLY BY ACB MAINTENANCE UTILITY
//*****************************************************************
//IMSACB   DD  DSN=IMS.ACBLIB,DISP=OLD
//SYSUT3   DD  UNIT=SYSDA,SPACE=(80,(100,100))
//SYSUT4   DD  UNIT=SYSDA,SPACE=(256,(100,100)),DCB=KEYLEN=30
//*****************************************************************
//* ACBGEN INPUT PARMS TO REBUILD ACBLIB
//*****************************************************************
//SYSIN    DD  *
         BUILD PSB=ALL
/*
//*****************************************************************
//* DATASETS USED ONLY BY IMS CATALOG POPULATE UTILITY
//*****************************************************************
//IMSACB01 DD  DSN=*.IMSACB,DISP=OLD      DO NOT REPLACE ASTERISK
//DFSVSAMP DD  .......      BUFFER POOL DEFINITIONS
//IEFRDER  DD  .......      LOG DATASET FOR CATALOG UPDATES
//IEFRDER2 DD  .......      LOG DATASET FOR CATALOG UPDATES
//*****************************************************************
//* LOAD INPUT PARMS FOR IMS CATALOG POPULATE UTILITY
//*****************************************************************
//DFS3PPRM DD  *
DLI,DFS3PU00,DFSCPL00,,,,,,,,,,Y,N,,,,,,,,,,DFSDF=CAT
/*
//
```

DD statements

ACBCATWK
Defines an optional work data set that contains a list of the ACB members that are written to the ACB library during ACB generation.

The ACBCATWK data set is an output data set for the ACB Maintenance utility and an input data set for the DFS3PU00 utility.

You must specify a single ACBCATWK data set. Multiple data sets are not supported by the utility.

Specify the ACBCATWK data set to improve the performance of the DFS3PU00 utility. The DFS3PU00 utility uses the list of names to determine which records in the IMS catalog need to be inserted or updated. If you do not specify the ACBCATWK data set, the DFS3PU00 utility processes all members in the ACB libraries that are referenced in the IMSACBxx DD statements.

DFSVSAMP
Defines the buffer pool parameters data set.

IEFRDER DD
Defines the primary IMS log data set.

IEFRDER2 DD
Defines the secondary IMS log data set.

IMS DD
Defines the IMS.PSBLIB and IMS.DBDLIB data sets.

IMSACB DD
Defines a single ACB library data set.

Restriction: This data set is modified and cannot be shared with other jobs.

IMSACB01
Defines an ACB library data set that contains the ACB members that are used to populate the IMS catalog. This DD statement is required.

This DD statement must specify the same data set defined in the IMSACB DD statement. To ensure that the same data set is referenced, code this DD statement with an asterisk as the high-level qualifier, as shown in the example: `//IMSACB01 DD DSN=*.ACBLIB,DISP=OLD`

IMSDSTAG DD
Optionally defines the IMS directory staging data set.

DFS3PPRM
Specifies execution parameters for the DFS3PU00 utility. The execution parameters specified by the DFS3PPRM DD statement include the following values:

- The PSB for the DFS3PU00 utility to use
- Whether DBRC is enabled
- Whether IRLM is enabled
- The name of the DFSDFxxx PROCLIB member that contains the IMS catalog attributes

If the DFS3PPRM DD statement is omitted, the DFS3UACB utility passes the following default execution parameters to the DFS3PU00 utility:

```
DLI,DFS3PU00,DFSCP001,,,,,,,,,,Y,N,,,,,,,,,,,'DFSDF=CAT'
```

The preceding default parameters run the DFS3PU00 utility in update mode with DBRC and without IRLM. The default values specify DFSDFCAT as the DFSDFxxx PROCLIB member.

To load an IMS catalog, you must specify the DFSCPL00 PSB in the execution parameters defined on the DFS3PPRM DD statement.

Attention: Specifying the DFSCPL00 PSB in the execution parameters of the DFS3PU00 utility deletes any existing IMS catalog before starting the load process.

If data sharing is enabled for your IMS catalog, you must specify IRLM support in the DFS3PPRM DD statement by specify Y in the IRLM support position and the IRLM ID in the following position.

The following example of the DFS3PPRM DD statement specifies the load PSB DFSCPL00, no IRLM support, and a DFSDFxxx member named DFSDF001:

```
//DFS3PPRM DD  *
DLI,DFS3PU00,DFSCPL00,,,,,,,,,,Y,N,,,,,,,,,,,DFSDF=001
```

In contrast to the preceding example, the following example of the DFS3PPRM DD statement specifies the update PSB DFSCP001, support by IRLM IRL1, and a DFSDFxxx member named DFSDF002:

```
//DFS3PPRM DD  *
DLI,DFS3PU00,DFSCP001,,,,,,,,,,Y,Y,IRL1,,,,,,,,,,DFSDF=002
```

PROCLIB DD
Defines the IMS.PROCLIB data set that contains the DFSDFxxx member that defines various attributes of the IMS catalog that are required by the utility.

SYSABEND DD
Defines the dump data set.

SYSIN DD
Defines the input control statement data sets. They can be on a tape volume, direct-access device, card reader, or be routed through the input stream. The input can be blocked as multiples of 80. During execution, this utility can process as many control statements as required.

SYSINP DD
An optional control statement sequential data set with 80-character fixed-length records. Only characters in positions 1 - 72 are read.

The control statement parameters, which are separated by blanks or commas, can be specified on one or more records.

For a desciption of the control statements that you can specify with the SYSINP DD statement, see “Catalog population control statements” on page 323.

SYSPRINT DD
Defines the output message data set.

When the SYSPRINT DD statement refers to a DASD or tape data set, you can control the block size of this data set with the BLKSIZE subparameter of the **DCB** parameter. If specified, the BLKSIZE value must be an exact multiple of 121 or a system ABEND013-20 can result. Omitting BLKSIZE from a DASD data set causes a system-determined block size to be used. Regardless of what value is specified for the LRECL parameter, the utility always uses a record length of 121.

SYSUT3 DD
Defines a work data set that is required if either PRECOMP or POSTCOMP is specified on the EXEC statement.

SYSUT4 DD
Same function as SYSUT3.

ACB generation control statements

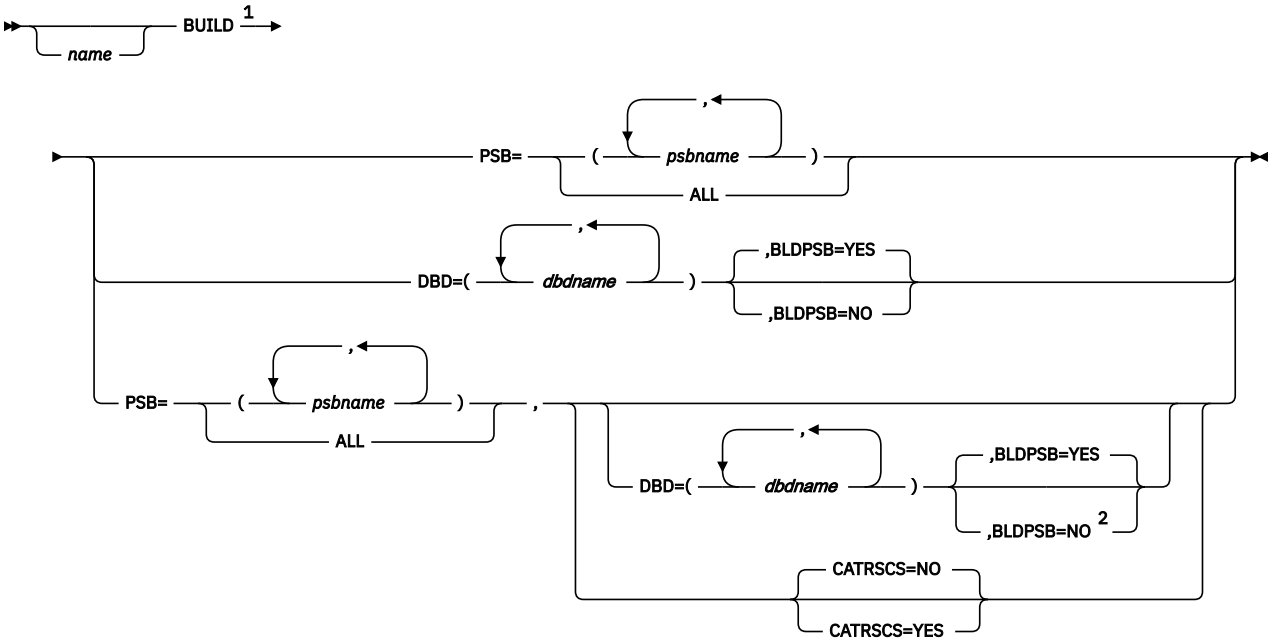
For the ACB generation phase of the DFS3UACB utility, you specify control statements in the utility JCL to specify the actions to take for the ACB members. You can specify BUILD statements and DELETE statements.

For BUILD statements, the DFS3UACB builds the specified ACB members and loads and inserts the corresponding records into the IMS catalog. For DELETE statements, the DFS3UACB utility deletes only the ACB members for the ACB library data set. No records are deleted from the IMS catalog. The removal of records from the IMS catalog is controlled by a retention policy.

The control statements must conform to the following guidelines:

- A statement is coded as a card image and is contained in columns 1 - 71.
- The control statement can optionally contain a name, starting in column 1.
- To continue a statement, enter a non-blank character in column 72 and begin the statement on the next line starting in column 16.
- The operation field must be preceded and followed by one or more blanks.
- The parameter is composed of one or more PSB or DBD names and must also be preceded and followed by one or more blanks.
- Commas, parentheses, and blanks can be used only as delimiting characters.
- Comments can be written following the last parameter of a control statement, separated from the parameter by one or more blanks.

DFS3UACB utility control syntax: BUILD format



Notes:

[1] There is no first in, first out (FIFO) process for the ACB Maintenance utility SYSIN input control statements. If both the BUILD PSB= and BUILD DBD= parameters are specified in the same application control block (ACB) generation job SYSIN control statement, DBD= operands are passed to the block builder utility program first. DFS0586I is issued if the DBD is not already in the ACBLIB data sets, regardless of where DBD= operands are entered in the SYSIN control statements.

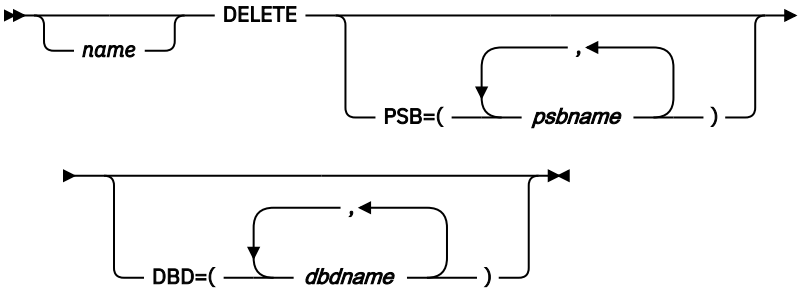
[2] If you specify the parameters PSB=ALL and BLDPSB=NO in the same statement, IMS builds all of the PSBs (BLDPSB=NO is ignored). Similarly, if you specify the BLDPSB=NO parameter for one DBD and the BLDPSB=YES parameter on another DBD in the same ACBGEN job, IMS builds all the PSBs that refer to the changed DBDs and ignores the BLDPSB=NO specification. Also, if you specify BLDPSB=NO, no PSBs will be built for all SYSIN ACBGEN control cards.

In the following example, all of the PSBs that are associated with the CUSTOMER and ORDER DBDs are rebuilt, even though BLDPSB=NO is specified for the CUSTOMER DBD:

```
BUILD DBD=(CUSTOMER),BLDPSB=NO
BUILD DBD=(ORDER),BLDPSB=YES
```

ACB Maintenance utility syntax: DELETE Format

The DELETE statement removes an ACB member from the ACB library, but does not remove the corresponding record from the IMS catalog.



ACB Maintenance utility parameters

BUILD
Specifies that blocks are built for the named PSBs, which refer to the named DBDs.

DELETE
Specifies that blocks are deleted from the ACBLIB data set. The named PSBs and all PSBs that refer to the named DBDs are deleted.

Deleting a block from the ACBLIB data set does not delete the corresponding record in the IMS catalog.

PSB=ALL
Specifies that blocks are built for all PSBs that currently reside in IMS.PSBLIB. You use this parameter to create an initial IMS.ACBLIB. When the PSB=ALL parameter is specified, all PSBs and DBDs (and any other modules) are deleted from the ACBLIB data set and their space is available for reuse. Then an ACB generation is executed for every PSB in the PSBLIB data set. Do not use this parameter with a DELETE statement.

Restriction: When you specify the BUILD PSB=ALL parameter on a SYSIN control statement, all PSBs must reside in a single PSBLIB data set. No concatenated PSBLIBs are recognized on the IMS DD statement.

Restriction: Specifying PSB=ALL with CATRSCS=NO (or defaulting to CATRSCS=NO) does not build or update the IMS catalog PSBs and DBDs. If some PSBs or DBDs are not built or updated, message DFS5008W is issued and those PSBs or DBDs are ignored.

PSB=(*psbname*)
Specifies that blocks are built or deleted for all PSBs that are named on this control statement. As many of this type of control statement as required can be submitted. This parameter adds a new PSB to IMS.ACBLIB or delete a PSB no longer in use. You can omit the parentheses if you supply a single parameter.

DBD=(*dbdname*)
Specifies that blocks are built or deleted for this DBD and for all PSBs that reference this DBD either directly or indirectly through logical relationships. The DBD to be built must already exist in the IMS.ACBLIB data set. The referencing PSBs must already exist in the IMS.ACBLIB data set. PSBs that are newly added to the IMS.PSBLIB data set must be referenced by PSB operands. Because deleting a PSB does not delete any DBDs referenced by the PSB, this parameter can be used to delete specific DBDs. However, deleting or building a DBD causes every PSB in the IMS.ACBLIB data set that references the named DBD to be rebuilt or deleted based on the request type. You can omit the parentheses if you supply a single parameter.

Example 1: PSB-a references DBD-a and DBD-b. A DBDGEN was done for DBD-a and DBD-b and the updated DBDs are in DBDLIB (but not ACBLIB yet). By specifying DBD-a in an ACB generation, DBD-a is rebuilt in ACBLIB and any referencing PSBs (in this case PSB-a) are also rebuilt. Even though PSB-a has been rebuilt, the ACBLIB is not usable because DBD-b was not specifically rebuilt in ACBLIB. For DBD-b to be rebuilt in ACBLIB, it must be explicitly specified in the ACB generation. Although the referencing PSB is completely updated, the updated DBDs must be explicitly specified in the ACB generation.

Every PSB processed by this program generates a member in the IMS.ACBLIB data set. DBDs referenced by PSBs generate a member the first time the specific DBD is processed or any time a DBD name appears on a control statement. All PSBs that reference the same DBD carry information in their directory entries to connect the PSB to the referenced DBDs.

Logical DBDs do not have members in IMS.ACBLIB and cannot be referenced on BUILD or DELETE control statements.

Example 2: The following examples illustrate uses of the BLDPSB parameter:

- The DBD named CUSTOMER was changed and all of the PSBs that refer to CUSTOMER need to be rebuilt:

```
BUILD DBD=CUSTOMER,BLDPSB=YES
```

- The DBDs named ORDER and INVENTORY are changed and all of the PSBs that refer to these DBDs need to be rebuilt:

```
BUILD DBD=(ORDER,INVENTORY),BLDPSB=YES
```

When a DBD is replaced in IMS.DBDLIB, it must also be included in a BUILD DBD control statement. This is the only valid way the DBD can be replaced in IMS.ACBLIB without doing a BUILD PSB=ALL.

If a BUILD PSB is performed that references a modified DBD on DBDLIB, the PSB replaced on ACBLIB will contain the updated version of the DBD. If this BUILD PSB occurs before a BUILD DBD for the changed DBD, ACBLIB will contain PSBs with different versions of the DBD. The PSBs specified in the BUILD PSB will contain the updated DBD, while those not built will reference the old DBD. When a DBD for a PSB on ACBLIB does not match the accessed database, the results will be unpredictable. (For example, U852 abend occurs because segment codes have been added or deleted in the changed DBD). Therefore, when DBDGEN is run for later use, do not build a PSB that refers to the changed DBD unless the database reflects the change.

When a physical DBD is changed and is referenced in a BUILD DBD statement, all physical DBDs that are logically related to the one that was changed (including primary indexes and secondary indexes) must also be referenced in a BUILD DBD statement. However, DBDs that are logically related to these DBDs do not need to be rebuilt.

The following figure illustrates the relationships between some physical databases, where A is the changed DBD. The following relationships exist:

- B and C are logically related to A.
- D is logically related to B.
- E is logically related to C.
- D and E are not referenced in the BUILD DBD statement because they are not logically related to A.

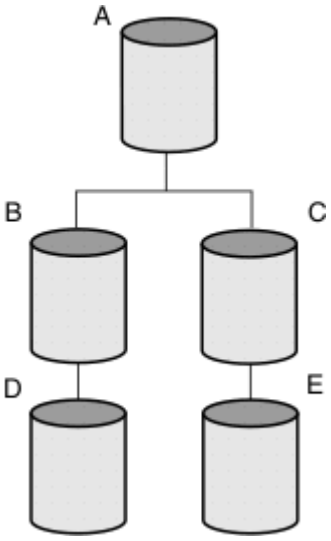


Figure 30. Example of logically related physical databases

BLDPSB=YES | NO

Specifies whether ACBGEN rebuilds all PSBs that reference a changed DBD in the BUILD DBD=(*dbdname*) statement.

YES

Indicates that ACBGEN rebuilds all PSBs that reference the changed DBD on the BUILD DBD=(*dbdname*) statement. The default is BLDPSB=YES.

NO

Indicates that ACBGEN does not rebuild PSBs that reference the changed DBD if the changed DBD does not change the physical structure of the database. For Fast Path DEDBs, the PSBs are rebuilt only when the number of segments, the number of fields within the segments of the database, or both are changed. For Fast Path MSDBs, the referencing PSBs are not rebuilt even if the database has physical structure changes.

CATRSCS=YES | NO
Specifies whether ACBGEN builds all PSBs and DBDs of the IMS catalog.

YES
Indicates that ACBGEN builds all IMS catalog PSBs and DBDs.

NO
Indicates that ACBGEN does not build any of the IMS catalog PSBs and DBDs. The default is CATRSCS=NO.

Catalog population control statements

The following control statement parameters, which are separated by blanks or commas, can be specified on one or more records by using the SYSINP DD statement.

DUPLIST
Specifies that the DFS3PU00 utility list each DBD or PSB resource in the input ACB library that is not added to the IMS catalog because it is a duplicate of an instance that is already in the IMS catalog. For each duplicate instance of a resource in the IMS catalog, the utility prints message DFS4436I.

If the MANAGEDACBS= statement is specified and the IMS management of ACBs is enabled, the utility also lists each DBD or PSB resources that is not added to the IMS directory or the staging data set because it is a duplicate of the instance of a resource that is already in the IMS directory.

If the UPDATE parameter is specified on the MANAGEDACBS= statement, the utility prints message DFS4531I for each duplicate instance.

If LATEST is specified or accepted as the default with the UPDATE parameter, the utility also prints message DFS4522I for each instance that is not added to the IMS directory because the instance of the resource in the IMS directory has a later timestamp.

If the STAGE parameter is specified on the MANAGEDACBS= statement, the utility prints message DFS4542I for each resource that is not copied to the staging data set because it is a duplicate instance.

If LATEST is specified or accepted as the default with the STAGE parameter, the utility also prints message DFS4539I for each instance that is not added to the staging data set because the instance of the resource in the IMS directory has a later timestamp.

ERRORMAX=*n*
Terminate the IMS Catalog Populate utility when more than *n* messages indicate errors that prevent certain DBDs and PSBs from having their metadata that is written to the IMS catalog. Duplicate instances of metadata do not count toward this limit. If this option is omitted, there is no limit.

RESOURCE_CHKP_FREQ=*n*
Specifies the number of DBD and PSB resource instances to be inserted between checkpoints. *n* can be a 1- to 8-digit numeric value of 1 to 99999999. The default value is 100.

SEGMENT_CHKP_FREQ=*n*
Specifies the number of segments to be inserted between checkpoints. When the number is reached, IMS finishes inserting all of the segments for the resource instance that is currently being processed before issuing the checkpoint. *n* can be a 1- to 8-digit numeric value of 1 to 99999999. The default value is 1000.

Note: The first checkpoint frequency number to be reached will cause a checkpoint to be taken and the counters will be reset to 0.

ISRTLIST

If the IMS management of ACBs is enabled, the utility also lists each DBD or PSB resources that is either added to the IMS directory or saved to the staging data set for importing into the IMS directory later.

The utility identifies the action taken for each resource by printing the following messages:

DFS4520I
The resource was added to the IMS directory as a new resource.

DFS4521I
This instance of the resource replaced an existing instance of the resource in the IMS directory.

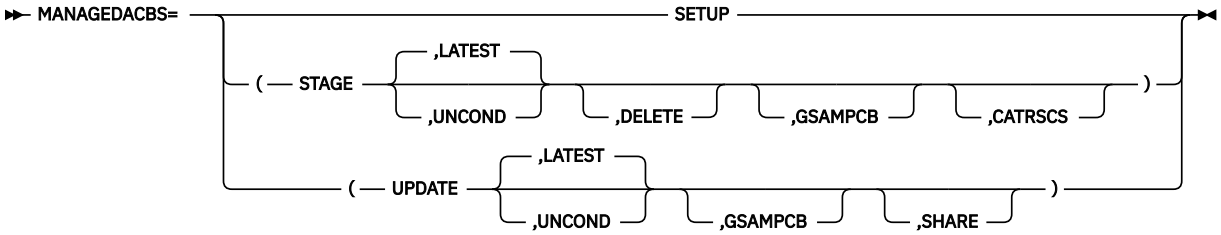
DFS4537I
The resource was copied to the staging data set. When it is imported into the IMS directory later, it will be inserted as a new resource in the IMS directory.

DFS4538I
The resource was copied to the staging data set. When it is imported into the IMS directory later, it will replace an existing instance of the resource in the IMS directory.

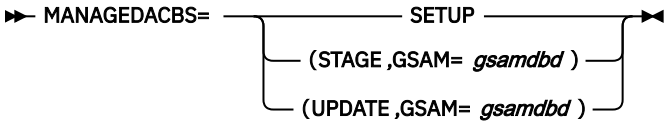
MANAGEDACBS=
Use the MANAGEDACBS control statement to perform the following actions:

- Set up IMS to manage the runtime application control blocks (ACBs) for your databases and program views.
- Update an IMS system that manages ACBs with new or modified ACBs from an ACB library data set.
- Save ACBs from an ACB library to a staging data set for later importing into an IMS system that manages ACBs.

The MANAGEDACBS statement can be specified according to the following syntax diagram:



If you use the *gsamdbd* parameter, the MANAGEDACBS statement can be specified according to the following syntax diagram:



The parentheses are required only if you specify multiple parameters on the MANAGEDACBS= statement.

The following list describes the parameters that you can specify on the MANAGEDACBS= statement:

SETUP
Creates the IMS directory data sets that are required by IMS to manage application control blocks (ACBs). Any existing instances of the IMS directory data sets are replaced.

Do not specify ACBMGMT=CATALOG in the CATALOG section of the DFSDFxxx PROCLIB member until after you successfully run the utility with MANAGEDACBS=SETUP specified. You must create the IMS directory and load it with the active ACBs before you enable the IMS management of ACBs.

When SETUP is specified, the utility inserts the ACBs that are in the input ACB libraries into the IMS directory data sets.

If the IMS catalog PSB DFSCP001 is specified in the utility JCL, the utility inserts any new or modified DBDs or PSBs into the IMS catalog.

If the IMS catalog PSB DFSCPL00 is specified in the utility JCL, the utility deletes all existing catalog records and reloads the IMS catalog.

When the utility adds an ACB to the IMS directory, it flags the corresponding DBD and PSB instances in the IMS catalog as active.

Do not run the DFS3PU00 utility as a BMP with **MANAGEDACBS=SETUP** specified.

STAGE

Saves the ACBs from the input ACB libraries to a staging data set. ACBs saved in the staging data set are not activated until you add them to the IMS system by issuing the IMPORT DEFN SOURCE(CATALOG) command.

When STAGE is specified, the IMS catalog PSB DFSCP001 must be specified in the utility JCL.

If the staging data set exists when the MANAGEDACBS=STAGE control statement is specified, the utility uses the existing staging data set. Any ACBs already in the staging data set are preserved. To scratch and re-create the staging data set when you stage new resources, specify MANAGEDACBS=(STAGE,DELETE). When DELETE is specified, any ACBs in the existing data set are lost.

When STAGE is specified, the DFS3PU00 utility copies all ACBs in the ACB libraries that do not already exist in the IMS system into the staging data set. If an ACB in the ACB library already exists in the IMS system, the utility saves it to the staging data set based on whether timestamp of the ACB meets the criteria of the UNCOND or LATEST parameter.

Do not specify STAGE when you do an initial load of the IMS catalog or before you enable the IMS management of ACBs. The STAGE parameter is for staging updates to existing IMS directory data sets only.

If you specify multiple parameters after MANAGEDACBS=, you must enclose them in parentheses. For example, (STAGE,LATEST). If you specify only STAGE, you can omit the parentheses.

CATRSCS
When you use the STAGE option with the CATRSCS option, IMS catalog resources are added to the staging directory data set.

DELETE
If the staging data set is not allocated to any online IMS system, scratch and recreate the staging data set before adding the resources to the staging data set. Any ACBs in the existing staging data set are lost.

GSAMPCB
GSAM resources are included for MANAGEDACBS= running in DLI mode using PSB DFSCP001. When GSAMPCB is specified, the IEFRDER batch log data set is not used by the catalog members information gather task. Only the catalog update task writes to the batch log data set for catalog database changes. GSAMPCB and DELETE are mutually exclusive.

LATEST
If an ACB already exists in the IMS system, do not save an instance of the ACB in an ACB library to the staging data set unless the instance in the ACB library has a later timestamp than the ACB in the IMS system.

This is the default.

UNCOND
If an ACB already exists in the IMS system, save an instance of the ACB in an ACB library to the staging data set unconditionally, unless the timestamp of the ACB in the ACB library is the same as the timestamp of the ACB in the IMS system.

gsamdbd

gsamdbd is the name of changed GSAM database. The specified database record is inserted into the IMS catalog, and the application control block is written to the staging or active directory based on the MANAGEDACBS control statement parameters that you specify.

You can use the *gsamdbd* variable with the STAGE or UPDATE parameter. However, LATEST, UNCOND, DELETE, SHARE, and GSAMPCB are not supported if you specify the *gsamdbd* variable.

Because the *gsamdbd* variable identifies a GSAM resource, the GSAMPCB batch logging rule applies only to the catalog update task that writes to the batch log data set for catalog database changes.

Follow these steps if you want to change GSAM DBDs by using the STAGE parameter and *gsamdbd*. This process writes only the GSAM DBD to the staging directory.

1. Change the GSAM DBD source.
2. Run a DBDGEN to generate the GSAM DBD into DBDLIB.
3. Run the DFS3PU00 utility with MANAGEDACBS=(STAGE,GSAM=*gsamdbd*).

   Tip: Only the DBDLIB DD card is required for the DFS3PU00 utility. ACBLIB and PSBLIB are not needed and are ignored if specified.

   The new GSAM DBD instance will be inserted to the catalog, and the application control block will be written to the staging directory.
4. Run the **IMPORT DEFN SOURCE(CATALOG)** or **IMPORT DEFN SOURCE(CATALOG) NAME(gsamdb)** command with or without OPTION(NOCHECK).

UPDATE

Updates existing IMS directory system data sets directly in exclusive mode. The ACBs are not placed in the staging data set.

Recommendation: Shut down all IMS systems that use the IMS directory when you specify the UPDATE parameter. When UPDATE is specified, the IMS Catalog Populate utility requires exclusive access to the IMS directory.

Also, the utility does not notify online IMS systems when it updates the IMS directory. Consequently, any IMS systems that are online when the utility updates the IMS directory must be restarted to load the updated ACBs.

When UPDATE is specified, the DFS3PU00 utility inserts ACBs that are not already in the IMS system into the IMS directory unconditionally. If an ACB exists in the IMS system, the instance of the ACB is replaced depending on the timestamp of the instance in the ACB library and the specification of the UNCOND or LATEST parameter.

Do not specify UPDATE when you do an initial load of the IMS catalog or before you enable the IMS management of ACBs. The UPDATE parameter is for updating existing IMS directory data sets only.

When UPDATE is specified, the IMS catalog PSB DFSCP001 must be specified in the utility JCL.

If you specify multiple parameters after MANAGEDACBS, you must enclose them in parentheses. For example, (UPDATE,LATEST). If you specify only UPDATE, you can omit the parentheses.

By default, the system data set names that are allocated for IMS management of ACBs are derived from the data set name prefix that is specified in the RECON data set when the first partition of the IMS catalog is defined. To override this default, you can specify the **SYSDSHLQ=** parameter in a TYPE=CATDSHLQ statement in the dynamic allocation (DFSMDA) macro.

LATEST

If an ACB already exists in the IMS system, do not replace it with an instance of the ACB from an ACB library unless the instance in the ACB library has a later timestamp than the ACB in the IMS system.

This is the default.

UNCOND

If an ACB already exists in the IMS system, replace it with an instance of the ACB from an ACB library unconditionally, unless the timestamp of the ACB in the ACB library is the same as the timestamp of the ACB in the IMS system.

SHARE
For dynamic option (DOPT) PSBs only, allocates the required IMS directory data sets in a shared mode so that DOPT PSBs can be added to the IMS catalog without interrupting online processing.

Do not specify SHARE if the utility is loading the IMS catalog, as indicated by the specification of DFSCPL00, the IMS catalog load PSB, in the execution parameters of the utility JCL.

Do not specify SHARE if any resources in the input ACBLIB data sets are not DOPT PSBs.

Before you run the utility with the SHARE option specified, the extended sharing of PDSE data sets must be enabled in your z/OS system. The extended sharing of PDSE data sets is enabled in z/OS by specifying PDSESHARING(EXTENDED) in the IGDSMS*xx* member in the SYS1.PARMLIB on each system in the sysplex. If extended PDSE sharing is not enabled when SHARE is specified, the utility allocates the required IMS catalog data sets in exclusive mode, which might cause conflicts with other IMS processes and application programs.

For more information about enabling extended PDSE sharing, see z/OS: Specifying Extended PDSE Sharing in a Multiple-System Environment.

GSAMPCB
GSAM resources are included for MANAGEDACBS= running in DLI mode using PSB DFSCP001. When GSAMPCB is specified, the IEFRDER batch log data set is not used by the catalog members information gather task. Only the catalog update task writes to the batch log data set for catalog database changes. GSAMPCB and DELETE are mutually exclusive.

gsamdbd

gsamdbd is the name of changed GSAM database. The specified database record is inserted into the IMS catalog, and the application control block is written to the staging or active directory based on the MANAGEDACBS control statement parameters that you specify.

You can use the *gsamdbd* variable with the STAGE or UPDATE parameter. However, LATEST, UNCOND, DELETE, SHARE, and GSAMPCB are not supported if you specify the *gsamdbd* variable.

Because the *gsamdbd* variable identifies a GSAM resource, the GSAMPCB batch logging rule applies only to the catalog update task that writes to the batch log data set for catalog database changes.

Follow these steps if you want to change GSAM DBDs by using the UPDATE parameter and *gsamdbd*:

1. Change the GSAM DBD source.
2. Run a DBDGEN to generate the GSAM DBD into DBDLIB.
3. Run the DFS3PU00 utility with MANAGEDACBS=(UPDATE,GSAM=*gsamdbd*).

   Tip: Only the DBDLIB DD card is required for the DFS3PU00 utility. ACBLIB and PSBLIB are not needed and are ignored if specified.

   The new GSAM DBD instance will be inserted to the catalog, and the application control block will be written to the IMS directory.

NODUPLIST
Do not print a list of resource instances that were not added. This parameter is the default.

NOISRTLIST
Do not print a list of inserted resource instances.

Return codes

The DFS3UACB utility returns the following codes:

0
Successful completion of all operations

4
One or more records were not loaded into the IMS catalog. Warning messages issued.

16
Program terminated due to severe errors

Related tasks

Populating the IMS catalog using the ACB Generation and Catalog Populate utility (DFS3UACB) (System Definition)

Related reference

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

“Application Control Blocks Maintenance utility (DFSUACB0)” on page 3

In IMS systems that use ACB libraries to manage runtime application control blocks, use the Application Control Blocks (ACB) Maintenance utility to save instruction execution and direct-access wait time and improve performance in application scheduling.

Chapter 7. IMS Catalog Alias Names utility (DFS3ALI0)

Use the IMS Catalog Alias Names utility (DFS3ALI0) to define multiple catalogs in an IMSPlex environment.

Overview

Use this utility to define a complete list of catalog aliases to IMS. This definition list is required so that IMS can perform catalog name translation on behalf of your user database PCBs.

You do not need to use this utility if your catalog database uses the standard prefix, DFSC.

Restrictions

No restrictions are documented for this utility.

Prerequisites

You must configure the DFSDF*xxx* member of the IMS.PROCLIB data set that is used in the start up JCL for the IMS system with a CATALOG section before you can use a catalog alias name.

Requirements

Your IMS system must use the IMS catalog.

Input and output

This utility takes a list of IMS catalog alias prefixes (4 characters each) and adds the names to the system DBD library.

JCL specifications

EXEC statement

```
//ALIAS  EXEC  PGM=DFS3ALI0
```

DD statements

JOBLIB/STEPLIB
The location of the IMS.SDFSRESL data set, which contains the executable modules for this utility.

DFSRESLB
The location of the IMS.SDFSRESL data set, which contains the executable modules for this utility. It also contains the catalog DFSCD000 DBD.

SYSPRINT
The data set that receives messages generated by the utility. The DCB parameters for the **SYSPRINT** data set are RECFM=FB and LRECL=121.

SYSLMOD
The location of the DBD library that the alias names are added to.

SYSIN
The data set that contains the input parameters for the utility. Include the complete list of catalog alias prefixes. Separate multiple entries with commas.

© Copyright IBM Corp. 1974, 2024

Sample JCL statements

```
//ALIAS  EXEC  PGM=DFS3ALI0
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSLMOD  DD  DSN=IMS.DBDLIB,DISP=OLD
//SYSLIN   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSIN    DD  *
prefix1,prefix2,…
/*
```

Related tasks

Defining an IMS catalog alias name (System Definition)

Sharing an IMS catalog (System Definition)

Related reference

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

Chapter 8. IMS Catalog Copy utilities (DFS3CCE0, DFS3CCI0)

Use the IMS Catalog Copy utilities to create a copy of an IMS catalog and, optionally, to create copies of the ACB, DBD, and PSB library data sets that were used to populate the IMS catalog.

If the IMS management of ACBs is enabled, you also can use the IMS Catalog Copy utilities to copy the active ACBs that are managed by the IMS catalog.

The IMS Catalog Copy utilities can be used for such purposes as migrating an IMS catalog database from a test environment to a production environment or for copying an IMS catalog database from a production environment to another installation for disaster recovery purposes.

The IMS Catalog Copy utilities include the IMS Catalog Export utility (DFS3CCE0) and the IMS Catalog Import utility (DFS3CCI0).

The Catalog Copy utilities can be run in either a batch region or as an online BMP.

Related concepts

IMS catalog data sets (System Definition)

Related tasks

Copying an IMS catalog (System Definition)

Related reference

"IMS Catalog Populate utility (DFS3PU00)" on page 373

Use the IMS Catalog Populate utility (DFS3PU00) to load, insert, or update DBD and PSB instances into the database data sets of the IMS catalog from ACB library data sets. If the IMS management of ACBs is enabled in the IMS system, the utility also adds the active application control blocks (ACBs) to the IMS directory data set that is managed by IMS.

z/OS: IEBCOPY (Library Copy) program

IMS Catalog Export utility (DFS3CCE0)

The IMS Catalog Export utility copies an IMS catalog and any included ACB, DBD, and PSB libraries to export data sets in the same job step.

In IMS-managed ACB environments, the IMS Catalog Export utility also can export the active ACBs that are managed by the IMS catalog.

To copy ACB, DBD, and PSB library data sets, the IMS Catalog Export utility calls the z/OS data set utility IEBCOPY, which does not support the concatenation of input data sets. So, only one of each type of library can be copied per job step.

The IMS Catalog Export utility produces an export statistics report when the utility copies an IMS catalog to an export data set. The IEBCOPY utility reports on its own statistics.

The IMS Catalog Export utility can be run in either a batch region or as an online BMP.

The IMS Catalog Export utility copies all of the DBD and PSB records in an IMS catalog. However, if an ACB library is specified on the IMSACB DD statement in the export JCL, the IMS Catalog Export utility copies only the segments in a DBD or PSB record that have a timestamp that matches the timestamp of the corresponding member in the specified ACB library. If the IMSACB DD statement is omitted, the utility copies all timestamp versions of the segments in every DBD or PSB record in the IMS catalog.

If an ACB library is specified on the IMSACB DD statement, but the ACB member for a particular DBD or PSB record in the IMS catalog is not found, the utility issues a warning message and copies all timestamp versions of the segments in the record.

Subsections:

© Copyright IBM Corp. 1974, 2024

- "Restrictions" on page 332
- "Prerequisites" on page 332
- "Requirements" on page 332
- "Recommendations" on page 332
- "Input and output" on page 332
- "Export JCL specifications" on page 334
- "Export DD statement descriptions" on page 336
- "Export statistics report" on page 337
- "Library statistics report" on page 339
- "Return codes" on page 339

Restrictions

The IMS Catalog Export utility uses the system service utility IEBCOPY to copy the ACB, PSB, and DBD libraries. The IEBCOPY utility does not support the concatenation of input data sets, so the IMS Catalog Export utility can copy only one data set for each type of library in a single job step.

Prerequisites

No prerequisites that are unique to the IMS Catalog Export utility are currently documented.

Requirements

No requirements that are unique to the IMS Catalog Export utility are currently documented.

Recommendations

No recommendations that are unique to the IMS Catalog Export utility are currently documented.

Input and output

To copy an IMS catalog, the primary input of the IMS Catalog Export utility (DFS3CCE0) is an IMS catalog. The utility uses the IMS catalog PSB DFSCP000 to read the IMS catalog database.

During the copying of the IMS catalog, the IMS Catalog Export utility must read the PSB and DBD libraries that are used to access the IMS catalog. For this purpose, an //IMS DD statement is required to reference the PSB and DBD libraries. A DD statement for the input IMS catalog data sets is not required.

When the IMS catalog that you are copying contains multiple timestamp versions of DBD and PSB members in the catalog records, you can specify the ACB library data set on the IMSACB DD statement to copy only the timestamp version of each DBD or PSB member in the catalog that matches the timestamp of the corresponding ACB library member.

The IMS Catalog Export utility uses the IMSACB DD statement to limit the number of timestamp versions of the DBD and PSB member segments that are copied from an IMS catalog.

To copy ACB, DBD, and PSB libraries, the data set for each must be specified as input by using the following DD statements:

- CCUACB DD statement for the ACB library data set
- CCUDBD DD statement for the DBD library data set
- CCUPSB DD statement for the PSB library data set

To maintain the integrity of the copy process, the data sets that are described by the CCUACB, CCUDBD, and CCUPSB DD statements should be the same as the data sets that are described by the IMS and IMSACB DD statements.

If the active ACBs of an IMS system are managed by using the ACB library, the output data sets for the exported copies of the IMS catalog, the ACB library, the DBD library, and the PSB library must be specified in the JCL by using the following DD statements:

- CCUCATEX DD statement for the IMS catalog export data set
- CCUACBEX DD statement for the ACB library export data set
- CCUDBDEX DD statement for the DBD library export data set
- CCUPSBEX DD statement for the PSB library export data set

If the IMS management of ACBs is enabled, use the CCUDIREX DD statement for the catalog system export data set that is required to copy the active ACBs managed by the IMS catalog. The catalog system export data set contains information about the active ACBs that are in the IMS catalog directory and the corresponding DBD and PSB records that are in the IMS catalog. If no active ACBs are found in the IMS catalog directory, the IMS catalog directory is not copied and exported.

If the CCUDIREX DD statement is used, the following DD statements are ignored:

- CCUACBEX DD
- CCUACB DD
- CCUDBDEX DD
- CCUDBD DD
- CCUPSBEX DD
- CCUPSB DD

The following figure shows the input and output for the IMS Catalog Export utility.

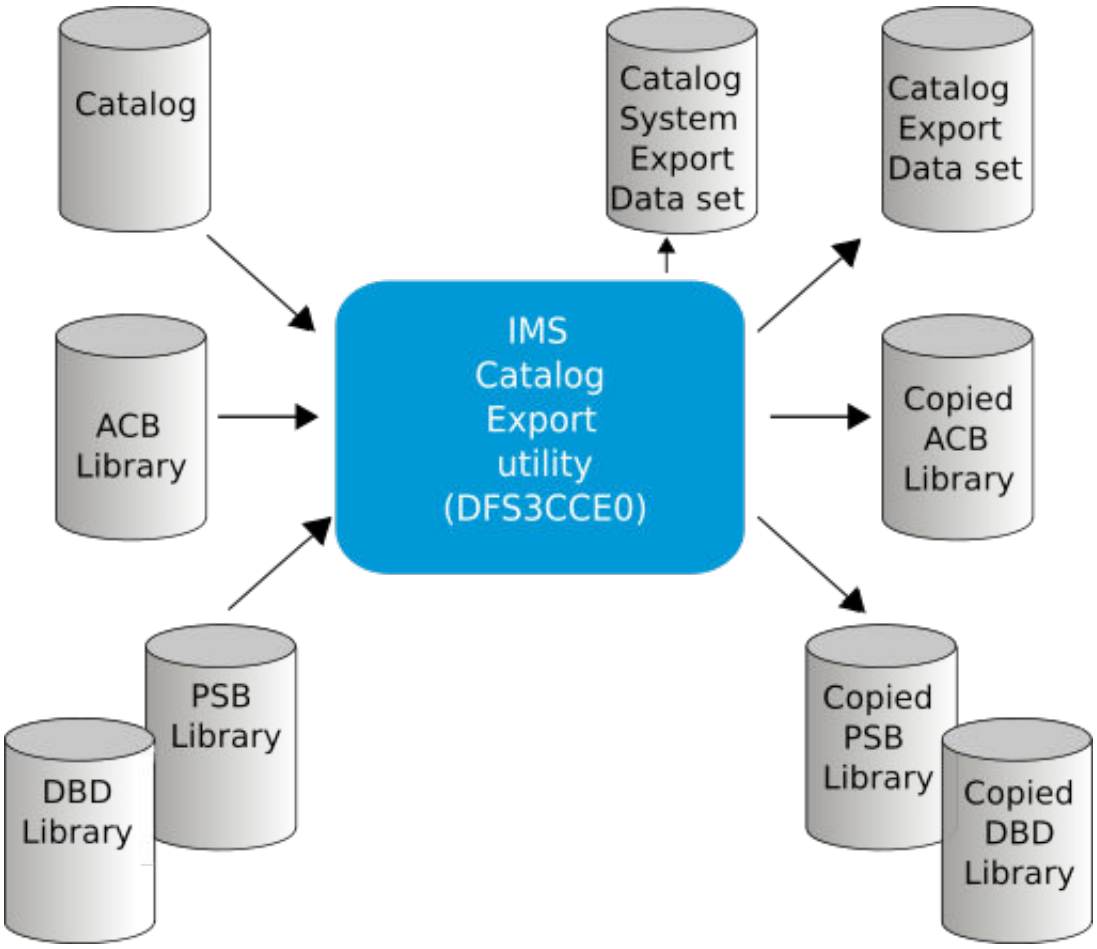


Figure 31. IMS Catalog Export utility input and output for export

Export JCL specifications

IMS Catalog Export utility JCL statements

The following JCL is an example of the JCL that you can use to copy an IMS catalog to an export data set.

```
//EXPRTCAT EXEC PGM=DFSRRC00,
// PARM=(DLI,DFS3CCE0,DFSCP000,,,,,,,,,,,Y,{Y|N},[irlmname],
//              ,,,,,,,,,,,'DFSDF=xxx')
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
//IMSACB   DD  ...            ACBLIB
//IMS      DD  ...            PSBLIB
//         DD  ...            DBDLIB
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//SYSABEND DD  SYSOUT=*       Dump data set
//SYSPRINT DD  SYSOUT=*       Messages
//IEFRDER  DD  ...            Log data set
//DFSVSAMP DD  ...            Buffer pool parameters
//CCUCATEX DD  ...            Catalog export data set
//CCUACB   DD  ...            Input data set for ACBLIB export function
//CCUACBEX DD  ...            Output data set for ACBLIB export function
//CCUDBD   DD  ...            Input data set for DBDLIB export function
//CCUDBDEX DD  ...            Output data set for DBDLIB export function
//CCUPSB   DD  ...            Input data set for PSBLIB export function
//CCUPSBEX DD  ...            Output data set for PSBLIB export function
```

If the IMS management of ACBs is enabled, the following JCL is an example of the JCL that you can use to copy an IMS catalog directory by using a catalog system export data set. If the CCUDIREX DD statement is used, the following statements are ignored:

- CCUACB DD
- CCUACBEX DD
- CCUDBD DD
- CCUDBDEX DD
- CCUPSB DD
- CCUPSBEX DD

```
//EXPRTCAT EXEC PGM=DFSRRC00,
// PARM=(DLI,DFS3CCE0,DFSCP000,,,,,,,,,,Y,{Y|N},[irlmname],
//                ,,,,,,,,,,'DFSDF=xxx')
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
//IMSACB   DD  ...             ACBLIB
//IMS      DD  ...             PSBLIB
//         DD  ...             DBDLIB
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//SYSABEND DD  SYSOUT=*        Dump data set
//SYSPRINT DD  SYSOUT=*        Messages
//IEFRDER  DD  ...             Log data set
//DFSVSAMP DD  ...             Buffer pool parameters
//CCUCATEX DD  ...             Catalog export data set
//CCUACB   DD  ...             Input data set for ACBLIB export function
//CCUACBEX DD  ...             Output data set for ACBLIB export function
//CCUDBD   DD  ...             Input data set for DBDLIB export function
//CCUDBDEX DD  ...             Output data set for DBDLIB export function
//CCUPSB   DD  ...             Input data set for PSBLIB export function
//CCUPSBEX DD  ...             Output data set for PSBLIB export function
//CCUDIREX DD  ...             Catalog directory export data set
```

If the catalog is not shared, the copy utility can access and copy it with a BMP job in an IMS that is using it. Here is the sample JCL job step for copying the catalog in a BMP:

```
//EXPRTCAT EXEC PGM=DFSRRC00,
//             PARM='BMP,DFS3CCE0,DFSCP000,,,,,,,,,,imsname'
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
//IMSACB   DD  ...             ACBLIB
//IMS      DD  ...             PSBLIB
//         DD  ...             DBDLIB
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//SYSABEND DD  SYSOUT=*        Dump data set
//SYSPRINT DD  SYSOUT=*        Messages
//CCUCATEX DD  ...             Catalog export data set
//CCUACB   DD  ...             Input data set for ACBLIB export function
//CCUACBEX DD  ...             Output data set for ACBLIB export function
//CCUDBD   DD  ...             Input data set for DBDLIB export function
//CCUDBDEX DD  ...             Output data set for DBDLIB export function
//CCUPSB   DD  ...             Input data set for PSBLIB export function
//CCUPSBEX DD  ...             Output data set for PSBLIB export function
```

If the IMS catalog directory data set is not shared, the copy utility can access and copy it with a BMP job in an IMS that is using it. Here is the sample JCL job step for copying the catalog in a BMP. If the CCUDIREX DD statement is used, the following statements are ignored:

- CCUACB DD
- CCUACBEX DD
- CCUDBD DD
- CCUDBDEX DD
- CCUPSB DD
- CCUPSBEX DD

```
//EXPRTCAT EXEC PGM=DFSRRC00,
//             PARM='BMP,DFS3CCE0,DFSCP000,,,,,,,,,,imsname'
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
//IMSACB   DD  ...             ACBLIB
//IMS      DD  ...             PSBLIB
//         DD  ...             DBDLIB
```

```
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//SYSABEND DD  SYSOUT=*       Dump data set
//SYSPRINT DD  SYSOUT=*       Messages
//CCUCATEX DD  ...            Catalog export data set
//CCUACB   DD  ...            Input data set for ACBLIB export function
//CCUACBEX DD  ...            Output data set for ACBLIB export function
//CCUDBD   DD  ...            Input data set for DBDLIB export function
//CCUDBDEX DD  ...            Output data set for DBDLIB export function
//CCUPSB   DD  ...            Input data set for PSBLIB export function
//CCUPSBEX DD  ...            Output data set for PSBLIB export function
//CCUDIREX DD  ...            Catalog directory export data set
```

Export DD statement descriptions

DFSRESLB DD
Points to an authorized library that contains the IMS SVC modules. For IMS batch, SDFSRESL and any data set that is concatenated to it on the DFSRESLB DD statement must be authorized through the Authorized Program Facility (APF). This DD statement provides an authorized library for the IMS SVC modules, which must be in an authorized library. The JOBLIB or STEPLIB statement does not need to be authorized for IMS batch.

DFSVSAMP
Defines the buffer pool parameters data set.

IEFRDER DD
Defines the primary IMS log data set.

CCUACB DD
For the IMS Catalog Export utility, defines the input ACB library data set to be copied.

The IMS Catalog Export utility does not support the concatenation of ACB library data sets on the CCUACB DD statement.

If both the CCUACB and IMSACB DD statements are specified in the JCL for the IMS Catalog Export utility, make sure that both statements point to the same ACB library data set.

CCUACBEX DD
Defines the output data set for the copy of the ACB library.

CCUCATEX DD
Defines the output data set for the copy of the IMS catalog.

CCUDBD DD
For the IMS Catalog Export utility, defines the input DBD library data set to be copied.

CCUDBDEX DD
Defines the output data set for the copy of the DBD library.

CCUDIREX DD

Defines the output data set to contain the information about the active ACBs of the IMS catalog. The information is required to populate a target IMS system with the active ACBs.

The data set name can be up to 44 characters long. However, the length of all qualifiers excluding the last qualifier can be up to 37 characters. The qualifiers are used as the high-level qualifiers for the exported IMS directory data sets.

CCUPSB DD
For the IMS Catalog Export utility, defines the input PSB library data set to be copied.

CCUPSBEX DD
Defines the output data set for the copy of the PSB library.

IMS DD
Defines the IMS.PSBLIB and IMS.DBDLIB data sets.

IMSACB DD
An optional DD statement that defines an ACB library data set that was used to populate the IMS catalog.

Specify the IMSACB DD statement to copy only the timestamp version of a DBD or PSB member in a record that matches the timestamp of the corresponding ACB library member. When the IMSACB DD statement is omitted, the IMS Catalog Export utility copies all timestamp versions of a member in each record.

If you need to define multiple ACB library data sets as input to the IMS Catalog Export utility, the ACB library data sets must be concatenated to the IMSACB DD statement.

The data sets referenced by the IMSACB DD statement are not copied. To copy ACB library data sets, you must reference them by the CCUACB DD statement and reference a corresponding output data set by the CCUACBEX DD statement.

If both the CCUACB and IMSACB DD statements are specified in the JCL for the IMS Catalog Export utility, make sure that both statements point to the same ACB library data set.

PROCLIB DD
Defines the IMS.PROCLIB data set that contains the DFSDFxxx member that defines various attributes of the IMS catalog that are required by the utility.

STEPLIB DD
Points to IMS.SDFSRESL, which contains the IMS nucleus and required IMS modules. If STEPLIB is unauthorized by having unauthorized libraries that are concatenated to IMS.SDFSRESL, you must include a DFSRESLB DD statement.

SYSABEND DD
Defines the dump data set.

SYSPRINT DD
Defines the output data set for the statistics report and utility messages. If a SYSPRINT data set is not defined, the utility defines one.

Export statistics report

The IMS Catalog Export utility outputs an export statistics report to the SYSPRINT data set.

The report has two sections. The first section lists the DBD or PSB records that the utility copies from the IMS catalog to the export data set. Each DBD or PSB record that is copied is listed as a separate entry with the record type, the DBD or PSB name, and the timestamp.

If the record for a DBD or PSB member in the IMS catalog contains multiple timestamp versions of the record segments, the record appears on multiple rows in the report, with each row representing a different timestamp version of the segments in the DBD or PSB record. The entries for the different timestamp versions of the segments in a DBD or PSB record are identical except for the timestamp, which reflects the time at which the corresponding version of the member was generated in the ACB library.

In the second section, the utility lists the total number of each type of segment that the utility copies to the export data set. The header segment is the root segment for a DBD or PSB member record. Regardless of whether the IMS catalog contains multiple timestamp versions of a DBD or PSB member, the IMS catalog contains only one root segment for each DBD or PSB name. The different timestamp versions of a DBD or PSB member are stored in the catalog as twin segments directly under the header segment.

In the following example, the IMS Catalog Export utility copied records for 24 DBD and PSB members. In the first section of the report, 26 rows are shown because the member records for DBD DBOVLFPC and PSB PSBCOMPT each have two timestamp versions in the IMS catalog.

In the second section of the example, the number of header segments reflects the actual number of DBD and PSB member names, while the sum of the DBD and PSB segments copied reflects the total number of timestamp versions that were copied.

```
DFS4803I DBD DBOHIDK5 WITH TIME STAMP 1133521555167 WAS COPIED TO THE EXPORT DATA
SET.
DFS4803I DBD DBOVLFPC WITH TIME STAMP 1133521555167 WAS COPIED TO THE EXPORT DATA
SET.
```

```
DFS4803I DBD DBOVLFPC WITH TIME STAMP 1133521561758 WAS COPIED TO THE EXPORT DATA
SET.
DFS4803I DBD DBVHDJ05 WITH TIME STAMP 1133521555167 WAS COPIED TO THE EXPORT DATA
SET.
DFS4803I DBD DFSCD000 WITH TIME STAMP 1133521563369 WAS COPIED TO THE EXPORT DATA
SET.
DFS4803I DBD DFSCX000 WITH TIME STAMP 1133521563369 WAS COPIED TO THE EXPORT DATA
SET.
DFS4803I DBD DHVNTZ02 WITH TIME STAMP 1133521561758 WAS COPIED TO THE EXPORT DATA
SET.
DFS4803I DBD DIVNTZ02 WITH TIME STAMP 1133521561758 WAS COPIED TO THE EXPORT DATA
SET.
DFS4803I DBD DLOHIDK5 WITH TIME STAMP 1133521550000 WAS COPIED TO THE EXPORT DATA
SET.
DFS4803I DBD DLVHDJ05 WITH TIME STAMP 1133521550000 WAS COPIED TO THE EXPORT DATA
SET.
DFS4803I DBD DLVNTZX2 WITH TIME STAMP 1133521560000 WAS COPIED TO THE EXPORT DATA
SET.
DFS4803I DBD DLVNTZ02 WITH TIME STAMP 1133521560000 WAS COPIED TO THE EXPORT DATA
SET.
DFS4803I DBD DXVNTZ02 WITH TIME STAMP 1133521561758 WAS COPIED TO THE EXPORT DATA
SET.
DFS4803I DBD D2XHDJ05 WITH TIME STAMP 1133521555167 WAS COPIED TO THE EXPORT DATA
SET.
DFS4803I DBD D2XHIDK5 WITH TIME STAMP 1133521555167 WAS COPIED TO THE EXPORT DATA
SET.
DFS4804I PSB CATCP000 WITH TIME STAMP 1133521563369 WAS COPIED TO THE EXPORT DATA
SET.
DFS4804I PSB DFSCPL00 WITH TIME STAMP 1133521563369 WAS COPIED TO THE EXPORT DATA
SET.
DFS4804I PSB DFSCP000 WITH TIME STAMP 1133521563369 WAS COPIED TO THE EXPORT DATA
SET.
DFS4804I PSB DFSCP001 WITH TIME STAMP 1133521563369 WAS COPIED TO THE EXPORT DATA
SET.
DFS4804I PSB DFSCP002 WITH TIME STAMP 1133521563369 WAS COPIED TO THE EXPORT DATA
SET.
DFS4804I PSB DFSCP003 WITH TIME STAMP 1133521563369 WAS COPIED TO THE EXPORT DATA
SET.
DFS4804I PSB PLVAPZ12 WITH TIME STAMP 1133521561758 WAS COPIED TO THE EXPORT DATA
SET.
DFS4804I PSB PLVAPZ22 WITH TIME STAMP 1133521561758 WAS COPIED TO THE EXPORT DATA
SET.
DFS4804I PSB PSBCOMPT WITH TIME STAMP 1133521555167 WAS COPIED TO THE EXPORT DATA
SET.
DFS4804I PSB PSBCOMPT WITH TIME STAMP 1133521561758 WAS COPIED TO THE EXPORT DATA
SET.
DFS4804I PSB PSBEJK05 WITH TIME STAMP 1133521555167 WAS COPIED TO THE EXPORT DATA
SET.

DFS4805I 00000024 HEADER SEGMENTS WERE COPIED TO THE EXPORT DATA SET.
DFS4805I 00000015 DBD SEGMENTS WERE COPIED TO THE EXPORT DATA SET.
DFS4805I 00000003 DBDRMK SEGMENTS WERE COPIED TO THE EXPORT DATA SET.
DFS4805I 00000010 DSET SEGMENTS WERE COPIED TO THE EXPORT DATA SET.
DFS4805I 00000329 SEGM SEGMENTS WERE COPIED TO THE EXPORT DATA SET.
DFS4805I 00000006 SEGMRMK SEGMENTS WERE COPIED TO THE EXPORT DATA SET.
DFS4805I 00000944 FLD SEGMENTS WERE COPIED TO THE EXPORT DATA SET.
DFS4805I 00000004 FLDRMK SEGMENTS WERE COPIED TO THE EXPORT DATA SET.
DFS4805I 00000944 MAR SEGMENTS WERE COPIED TO THE EXPORT DATA SET.
DFS4805I 00000038 LCHILD SEGMENTS WERE COPIED TO THE EXPORT DATA SET.
DFS4805I 00000003 XDFLD SEGMENTS WERE COPIED TO THE EXPORT DATA SET.
DFS4805I 00000011 PSB SEGMENTS WERE COPIED TO THE EXPORT DATA SET.
DFS4805I 00000004 PSBRMK SEGMENTS WERE COPIED TO THE EXPORT DATA SET.
DFS4805I 00000036 PCB SEGMENTS WERE COPIED TO THE EXPORT DATA SET.
DFS4805I 00000009 PCBRMK SEGMENTS WERE COPIED TO THE EXPORT DATA SET.
DFS4805I 00000860 SS SEGMENTS WERE COPIED TO THE EXPORT DATA SET.
DFS4805I 00000030 DBDXREF SEGMENTS WERE COPIED TO THE EXPORT DATA SET.
```

If you used the CCUDIREX DD statement to copy the active ACBs of the IMS catalog, a list is generated of the output data sets that are created and that must be available to the system where the IMS Catalog Import utility will be used. The list of the output data sets is printed at the end of the export job output.

The following example shows three output data sets that must be available for the IMS Catalog Import utility:

```
CATALOG COPY EXPORT UTILITY

DFS4813I THE FOLLOWING DATA SETS MUST BE MADE AVAILABLE FOR
IMPORT OF THIS IMS CATALOG DIRECTORY:

  IMSTESTS.DIR.CONTROL
  IMSTESTS.DIR.DX1001
  IMSTESTS.DIR.DX1002
```

Library statistics report

The IMS Catalog Export utility calls the z/OS data set utility IEBCOPY to copy ACB, DBD, and PSB libraries. The IEBCOPY utility provides statistics and messages for the copy operation. For more information about the IEBCOPY utility, see the related link near the bottom of this topic.

Return codes

The IMS Catalog Export utility returns the following codes:

0
Successful completion of all operations

4
One or more warning messages were issued.

8 or greater
Program terminated due to severe errors

IMS Catalog Import utility (DFS3CCI0)

Use IMS Catalog Import utility (DFS3CCI0) at the destination environment to load or update an IMS catalog and copy any included ACB, DBD, and PSB libraries from the export data sets into their destination data sets.

If the IMS management of ACBs is enabled in the originating environment, you can use the IMS Catalog Import utility to import the active ACBs that are in the IMS catalog directory data set. The IMS Catalog Import utility can install the ACBs into an IMS catalog directory at the target environment.

To copy ACB, DBD, and PSB library data sets, the IMS Catalog Import utility calls the z/OS data set utility IEBCOPY, which does not support the concatenation of input data sets. So, only one of each type of library can be copied per job step.

The IMS Catalog Import utility produces an import statistics report when it loads or updates the IMS catalog at the destination environment. The IEBCOPY utility reports on its own statistics.

The IMS Catalog Import utility can be run in either a batch region or as an online BMP.

Subsections:

- “Restrictions” on page 340
- “Prerequisites” on page 340
- “Requirements” on page 340
- “Recommendations” on page 340
- “Input and output” on page 340
- “Import JCL specifications” on page 343
- “Import DD statement descriptions” on page 344
- “Control statements” on page 346
- “Import statistics report” on page 350

- “Library statistics report” on page 352
- “Return codes” on page 352

Restrictions

The IMS Catalog Import utility uses the system service utility IEBCOPY to copy the ACB, PSB, and DBD libraries. The IEBCOPY utility does not support the concatenation of input data sets, so the IMS Catalog Import utility can copy only one data set for each type of library in a single job step.

Prerequisites

No prerequisites that are unique to the IMS Catalog Import utility are currently documented.

Requirements

No requirements that are unique to the IMS Catalog Import utility are currently documented.

Recommendations

No recommendations that are unique to the IMS Catalog Import utility are currently documented.

Input and output

The primary input for the IMS Catalog Import utility (DFS3CCI0) is the export data set that was generated by the IMS Catalog Export utility (DFS3CCE0) and that contains the copy of the IMS catalog. A CCUCATIM DD statement is required to identify this data set. If IMS manages ACBs at the destination IMS environment, another key input is the export data set that contains information about the IMS-managed active ACBs from the originating IMS environment. The CCUDIRIM DD statement is required to identify this data set.

During the copying of the IMS catalog, the IMS Catalog Export utility must read the PSB and DBD libraries that are used to access the IMS catalog. For this purpose, an //IMS DD statement is required to reference the PSB and DBD libraries. A DD statement for the input IMS catalog data sets is not required.

The IMS Catalog Copy utilities use the IMSACB DD statement to limit the number of timestamp versions of the DBD and PSB member segments that are copied from an IMS catalog.

To maintain the integrity of the copy process, the data sets that are described by the CCUACB, CCUDBD, and CCUPSB DD statements should be the same as the data sets that are described by the IMS and IMSACB DD statements.

If ACB libraries, DBD libraries, and PSB libraries were copied by the IMS Catalog Export utility, identify the libraries in the import JCL by using the following DD statements:

- CCUACBIM DD statement for the ACB library export data set
- CCUDBDIM DD statement for the DBD library export data set
- CCUPSBIM DD statement for the PSB library export data set

To use the IMS Catalog Import utility to copy the ACB libraries, DBD libraries, and PSB libraries to destination libraries, identify the destination libraries in the import JCL by using the following DD statements:

- CCUACB DD statement for the ACB library data set
- CCUDBD DD statement for the DBD library data set
- CCUPSB DD statement for the PSB library data set

If the IMS-managed active ACBs from the IMS directory were copied by the IMS Catalog Export utility, in the import JCL identify the catalog system export data set for the ACBs by using the CCUDIRIM DD statement. The CCUDIRIM DD statement also tells the utility that the ACBs are being added to an IMS directory data set in an IMS-managed ACB environment.

If the CCUDIRIM DD statement is used, the following DD statements are ignored if they are specified:

- CCUACBIM DD
- CCUACB DD
- CCUDBDIM DD
- CCUDBD DD
- CCUPSBIM DD
- CCUPSB DD

Important: If you use the CCUDIRIM DD statement, you must use the SYSINP DD statement in the JCL of the IMS Catalog Import utility to specify the MANAGEDACBS= control statement parameter. The MANAGEDACBS= control statement parameter directs the IMS Catalog Import utility to either create or update the IMS directory data set of the target system with the copied ACBs. For a description of the MANAGEDACBS= control statement parameter, see "Control statements" on page 346.

The IMS Catalog Import utility uses the CCUDIRIM DD statement to copy only the segments that represent the active ACBs into the IMS directory at the destination environment.

The following figure shows the input and output for the IMS Catalog Import utility.

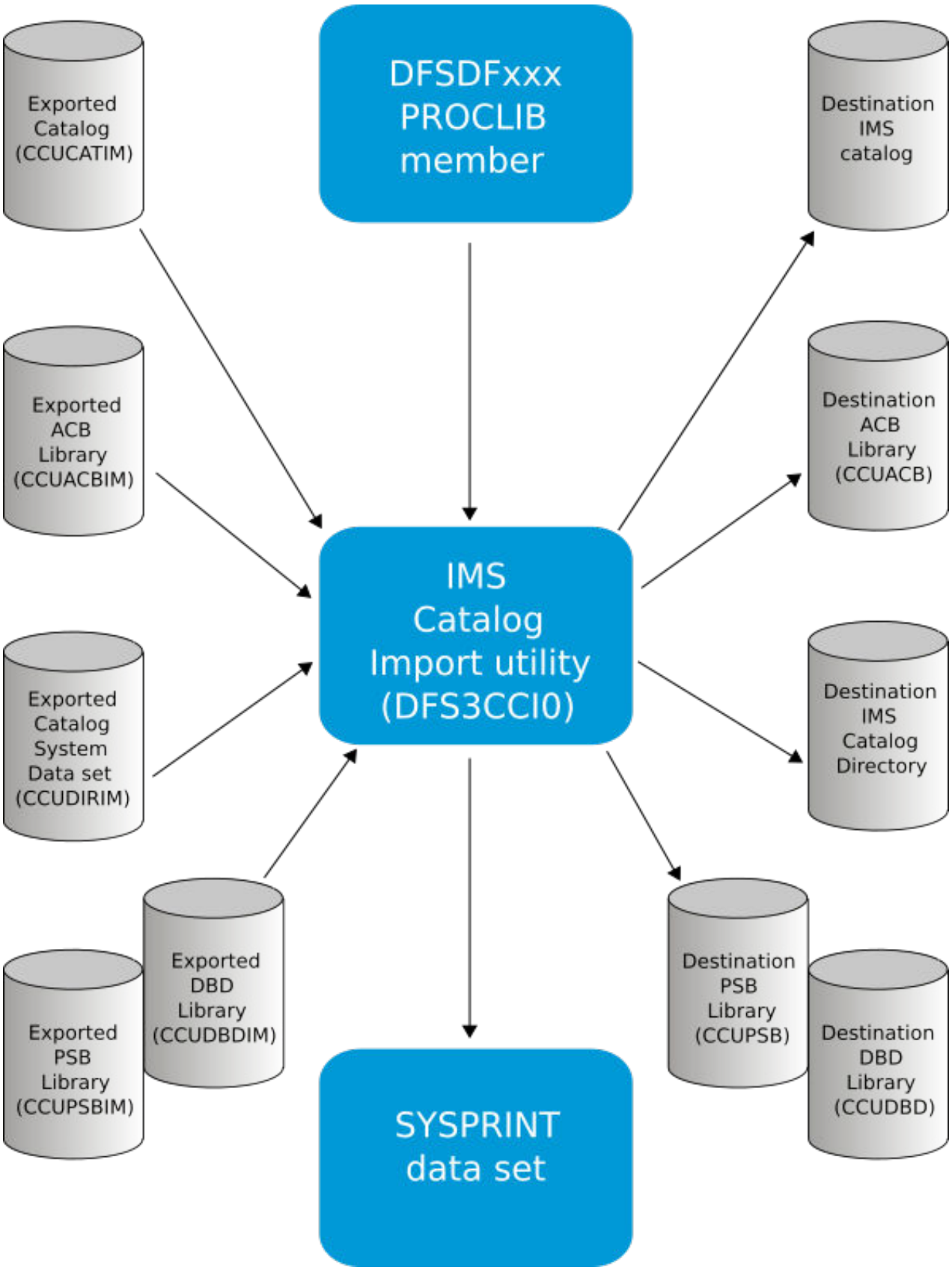


Figure 32. IMS Catalog Import utility input and output for import

Import JCL specifications

JCL statements for importing to an existing IMS catalog in update mode

The following sample JCL statements can be used to import exported DBD and PSB metadata to an existing catalog or to a new catalog. Specified ACB, DBD, and PSB libraries can be copied to the specified destination ACB, DBD, and PSB libraries.

```
//IMPRTCAT EXEC PGM=DFS3CCI0,
// PARM=(DLI,DFS3CCI0,DFSCP001,,,,,,,,,,Y,{Y|N},[irlmname],
//             ,,,,,,,,,,'DFSDF=xxx')
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//SYSABEND DD  SYSOUT=*       Dump data set
//SYSPRINT DD  SYSOUT=*       Messages
//IEFRDER  DD  ...            Log data set with catalog updates
//DFSVSAMP DD  ...            Buffer pool parameters
//IMSACB   DD  ...            ACBLIB
//IMS      DD  ...            PSBLIB
//         DD  ...            DBDLIB
//CCUCATIM DD  ...            Catalog import data set
//CCUACBIM DD  ...            Input data set for ACBLIB import function
//CCUACB   DD  ...            Output data set for ACBLIB import function
//CCUDBDIM DD  ...            Input data set for DBDLIB import function
//CCUDBD   DD  ...            Output data set for DBDLIB import function
//CCUPSBIM DD  ...            Input data set for PSBLIB import function
//CCUPSB   DD  ...            Output data set for PSBLIB import function
```

If the catalog is not shared, the copy utility can update it with a BMP job in an IMS that is using it. Here is the sample JCL job step for updating the catalog in a BMP:

```
//IMPRTCAT EXEC PGM=DFS3CCI0,
//              PARM=(BMP,DFS3CCI0,DFSCP001,,,,,,,,,,imsname'
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//SYSABEND DD  SYSOUT=*       Dump data set
//SYSPRINT DD  SYSOUT=*       Messages
//IEFRDER  DD  ...            Log data set with catalog updates
//DFSVSAMP DD  ...            Buffer pool parameters
//IMSACB   DD  ...            ACBLIB
//IMS      DD  ...            PSBLIB
//         DD  ...            DBDLIB
//CCUCATIM DD  ...            Catalog import data set
//CCUACBIM DD  ...            Input data set for ACBLIB import function
//CCUACB   DD  ...            Output data set for ACBLIB import function
//CCUDBDIM DD  ...            Input data set for DBDLIB import function
//CCUDBD   DD  ...            Output data set for DBDLIB import function
//CCUPSBIM DD  ...            Input data set for PSBLIB import function
//CCUPSB   DD  ...            Output data set for PSBLIB import function
```

The following sample JCL statements can be used to import the IMS catalog directory from the originating environment to an existing IMS catalog at the destination environment that already manages active ACBs.

```
//IMPRTCAT EXEC PGM=DFS3CCI0,
// PARM=(DLI,DFS3CCI0,DFSCP001,,,,,,,,,,Y,{Y|N},[irlmname],
//             ,,,,,,,,,,'DFSDF=xxx')
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//SYSABEND DD  SYSOUT=*       Dump data set
//SYSPRINT DD  SYSOUT=*       Messages
//IEFRDER  DD  ...            Log data set with catalog updates
//DFSVSAMP DD  ...            Buffer pool parameters
//CCUCATIM DD  ...            Catalog import data set
//CCUDIRIM DD  ...            Catalog directory import data set
//SYSINP   DD  *              MANAGEDACBS=UPDATE
```

JCL statements for importing to a new IMS catalog in load mode

The following JCL is an example of the JCL that you can use to load an IMS catalog from an export data set.

If the database data sets of the IMS catalog do not exist, IMS creates them automatically during execution of the IMS Catalog Import utility.

If you want to control data set size or placement yourself, you can use the IMS Catalog Populate utility (DFS3PU00) to evaluate the DASD space required for the data sets of the IMS catalog. For more information, see “Import statistics report” on page 350.

After the IMS catalog is updated or loaded, back up the IMS catalog by using GENJCL.IC or standard image copy JCL.

```
//IMPRTCAT EXEC PGM=DFS3CCI0,
// PARM=(DLI,DFS3CCI0,DFSCPL00,,,,,,,,,,Y,{Y|N},[irlmname],
//              ,,,,,,,,,,'DFSDF=xxx')
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//SYSABEND DD  SYSOUT=*       Dump data set
//SYSPRINT DD  SYSOUT=*       Messages
//IEFRDER  DD  ...            Log data set with catalog updates
//DFSVSAMP DD  ...            Buffer pool parameters
//IMSACB   DD  ...            ACBLIB
//IMS      DD  ...            PSBLIB
//         DD  ...            DBDLIB
//CCUCATIM DD  ...            Catalog import data set
//CCUACBIM DD  ...            Input data set for ACBLIB import function
//CCUACB   DD  ...            Output data set for ACBLIB import function
//CCUDBDIM DD  ...            Input data set for DBDLIB import function
//CCUDBD   DD  ...            Output data set for DBDLIB import function
//CCUPSBIM DD  ...            Input data set for PSBLIB import function
//CCUPSB   DD  ...            Output data set for PSBLIB import function
```

The following JCL is an example of the JCL that you can use to load an IMS directory from a catalog system export data set.

```
//IMPRTCAT EXEC PGM=DFS3CCI0,
// PARM=(DLI,DFS3CCI0,DFSCPL00,,,,,,,,,,Y,{Y|N},[irlmname],
//              ,,,,,,,,,,'DFSDF=xxx')
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//SYSABEND DD  SYSOUT=*       Dump data set
//SYSPRINT DD  SYSOUT=*       Messages
//IEFRDER  DD  ...            Log data set with catalog updates
//DFSVSAMP DD  ...            Buffer pool parameters
//CCUCATIM DD  ...            Catalog import data set
//CCUDIRIM DD  ...            Catalog directory import data set
//SYSIN    DD  *              MANAGEDACBS=SETUP
```

Import DD statement descriptions

DFSRESLB DD

Points to an authorized library that contains the IMS SVC modules. For IMS batch, SDFSRESL and any data set that is concatenated to it on the DFSRESLB DD statement must be authorized through the Authorized Program Facility (APF). This DD statement provides an authorized library for the IMS SVC modules, which must be in an authorized library. The JOBLIB or STEPLIB statement does not need to be authorized for IMS batch.

DFSVSAMP

Defines the buffer pool parameters data set.

IEFRDER DD

Defines the primary IMS log data set.

CCUACB DD

For the IMS Catalog Import utility, defines the destination ACB library data set.

CCUACBIM DD
Defines the input data set that holds the copy of the ACB library to be imported.

CCUCATIM DD
Defines the input data set that holds the copy of the IMS catalog to be imported.

CCUDBD DD
For the IMS Catalog Import utility, defines the destination DBD library data set.

CCUDBDIM DD
Defines the input data set that holds the copy of the DBD library to be imported.

CCUDIRIM DD
Defines the input data set that holds information about the active ACBs of the IMS catalog to be imported.

CCUPSB DD
For the IMS Catalog Import utility, defines the destination PSB library data set.

CCUPSBIM DD
Defines the input data set that holds the copy of the PSB library to be imported.

IMS DD
Defines the IMS.PSBLIB and IMS.DBDLIB data sets.

IMSACB DD
An optional DD statement that defines the input data set that holds the copy of the ACB library used to populate the IMS catalog at the originating IMS system.

If multiple ACB library data sets are defined as input, the ACB library data sets must be concatenated to the IMSACB DD statement.

The data sets referenced by the IMSACB DD statement are not imported. To import ACB library data sets, you must reference them by using the CCUACBIM DD statement and reference a corresponding destination data set by the CCUACB DD statement.

PROCLIB DD
Defines the IMS.PROCLIB data set that contains the DFSDFxxx member that defines various attributes of the IMS catalog that are required by the utility.

STEPLIB DD
Points to IMS.SDFSRESL, which contains the IMS nucleus and required IMS modules. If STEPLIB is unauthorized by having unauthorized libraries that are concatenated to IMS.SDFSRESL, you must include a DFSRESLB DD statement.

SYSABEND DD
Defines the dump data set.

SYSINP DD

If the CCUDIRIM DD statement is used, a required control statement sequential data set with 80-character fixed-length records. Only characters in positions 1 - 72 are read.

The control statement parameters, which are separated by blanks or commas, can be specified on one or more records.

If you specify the CCUDIRIM DD statement, you must specify the MANAGEDACBS= control statement parameter by using the SYSINP DD statement in the JCL of the IMS Catalog Import utility. The MANAGEDACBS= control statement parameter directs the IMS Catalog Import utility to either create or update the IMS directory data set of the target system with the copied ACBs.

For a description of the control statements that you can specify with the SYSINP DD statement, see “Control statements” on page 346.

SYSPRINT DD
Defines the output data set for the statistics report and utility messages. If a SYSPRINT data set is not defined, the utility defines one.

Control statements

The following control statement parameters, which are separated by blanks or commas, can be specified on one or more records by using the SYSINP DD statement.

DUPLIST
Specifies that the DFS3PU00 utility list each DBD or PSB resource in the input ACB library that is not added to the IMS catalog because it is a duplicate of an instance that is already in the IMS catalog. For each duplicate instance of a resource in the IMS catalog, the utility prints message DFS4436I.

If the MANAGEDACBS= statement is specified and the IMS management of ACBs is enabled, the utility also lists each DBD or PSB resources that is not added to the IMS directory or the staging data set because it is a duplicate of the instance of a resource that is already in the IMS directory.

If the UPDATE parameter is specified on the MANAGEDACBS= statement, the utility prints message DFS4531I for each duplicate instance.

If LATEST is specified or accepted as the default with the UPDATE parameter, the utility also prints message DFS4522I for each instance that is not added to the IMS directory because the instance of the resource in the IMS directory has a later timestamp.

If the STAGE parameter is specified on the MANAGEDACBS= statement, the utility prints message DFS4542I for each resource that is not copied to the staging data set because it is a duplicate instance.

If LATEST is specified or accepted as the default with the STAGE parameter, the utility also prints message DFS4539I for each instance that is not added to the staging data set because the instance of the resource in the IMS directory has a later timestamp.

ERRORMAX=*n*
Terminate the IMS Catalog Populate utility when more than *n* messages indicate errors that prevent certain DBDs and PSBs from having their metadata that is written to the IMS catalog. Duplicate instances of metadata do not count toward this limit. If this option is omitted, there is no limit.

RESOURCE_CHKP_FREQ=*n*
Specifies the number of DBD and PSB resource instances to be inserted between checkpoints. *n* can be a 1- to 8-digit numeric value of 1 to 99999999. The default value is 100.

SEGMENT_CHKP_FREQ=*n*
Specifies the number of segments to be inserted between checkpoints. When the number is reached, IMS finishes inserting all of the segments for the resource instance that is currently being processed before issuing the checkpoint. *n* can be a 1- to 8-digit numeric value of 1 to 99999999. The default value is 1000.

Note: The first checkpoint frequency number to be reached will cause a checkpoint to be taken and the counters will be reset to 0.

ISRTLIST

If the IMS management of ACBs is enabled, the utility also lists each DBD or PSB resources that is either added to the IMS directory or saved to the staging data set for importing into the IMS directory later.

The utility identifies the action taken for each resource by printing the following messages:

DFS4520I
The resource was added to the IMS directory as a new resource.

DFS4521I
This instance of the resource replaced an existing instance of the resource in the IMS directory.

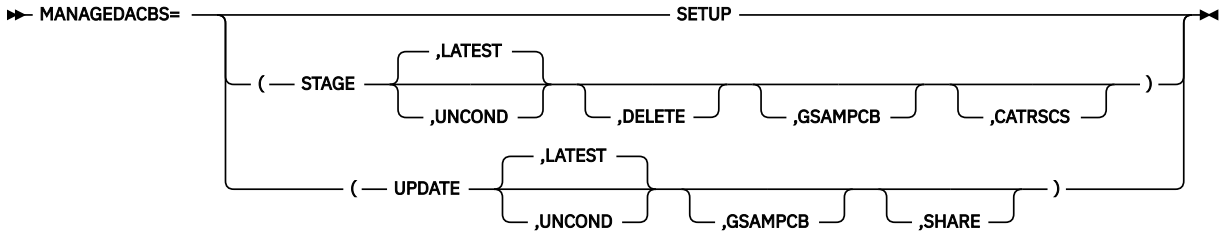
DFS4537I
The resource was copied to the staging data set. When it is imported into the IMS directory later, it will be inserted as a new resource in the IMS directory.

DFS4538I
The resource was copied to the staging data set. When it is imported into the IMS directory later, it will replace an existing instance of the resource in the IMS directory.

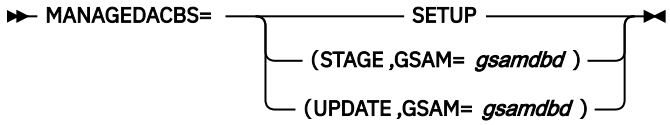
MANAGEDACBS=
Use the MANAGEDACBS control statement to perform the following actions:

- Set up IMS to manage the runtime application control blocks (ACBs) for your databases and program views.
- Update an IMS system that manages ACBs with new or modified ACBs from an ACB library data set.
- Save ACBs from an ACB library to a staging data set for later importing into an IMS system that manages ACBs.

The MANAGEDACBS statement can be specified according to the following syntax diagram:



If you use the *gsamdbd* parameter, the MANAGEDACBS statement can be specified according to the following syntax diagram:



The parentheses are required only if you specify multiple parameters on the MANAGEDACBS= statement.

The following list describes the parameters that you can specify on the MANAGEDACBS= statement:

SETUP
Creates the IMS directory data sets that are required by IMS to manage application control blocks (ACBs). Any existing instances of the IMS directory data sets are replaced.

Do not specify ACBMGMT=CATALOG in the CATALOG section of the DFSDFxxx PROCLIB member until after you successfully run the utility with MANAGEDACBS=SETUP specified. You must create the IMS directory and load it with the active ACBs before you enable the IMS management of ACBs.

When SETUP is specified, the utility inserts the ACBs that are in the input ACB libraries into the IMS directory data sets.

If the IMS catalog PSB DFSCP001 is specified in the utility JCL, the utility inserts any new or modified DBDs or PSBs into the IMS catalog.

If the IMS catalog PSB DFSCPL00 is specified in the utility JCL, the utility deletes all existing catalog records and reloads the IMS catalog.

When the utility adds an ACB to the IMS directory, it flags the corresponding DBD and PSB instances in the IMS catalog as active.

Do not run the DFS3PU00 utility as a BMP with **MANAGEDACBS=SETUP** specified.

STAGE

Saves the ACBs from the input ACB libraries to a staging data set. ACBs saved in the staging data set are not activated until you add them to the IMS system by issuing the IMPORT DEFN SOURCE(CATALOG) command.

When STAGE is specified, the IMS catalog PSB DFSCP001 must be specified in the utility JCL.

If the staging data set exists when the MANAGEDACBS=STAGE control statement is specified, the utility uses the existing staging data set. Any ACBs already in the staging data set are preserved. To scratch and re-create the staging data set when you stage new resources, specify MANAGEDACBS=(STAGE,DELETE). When DELETE is specified, any ACBs in the existing data set are lost.

When STAGE is specified, the DFS3PU00 utility copies all ACBs in the ACB libraries that do not already exist in the IMS system into the staging data set. If an ACB in the ACB library already exists in the IMS system, the utility saves it to the staging data set based on whether timestamp of the ACB meets the criteria of the UNCOND or LATEST parameter.

Do not specify STAGE when you do an initial load of the IMS catalog or before you enable the IMS management of ACBs. The STAGE parameter is for staging updates to existing IMS directory data sets only.

If you specify multiple parameters after MANAGEDACBS=, you must enclose them in parentheses. For example, (STAGE,LATEST). If you specify only STAGE, you can omit the parentheses.

CATRSCS
When you use the STAGE option with the CATRSCS option, IMS catalog resources are added to the staging directory data set.

DELETE
If the staging data set is not allocated to any online IMS system, scratch and recreate the staging data set before adding the resources to the staging data set. Any ACBs in the existing staging data set are lost.

GSAMPCB
GSAM resources are included for MANAGEDACBS= running in DLI mode using PSB DFSCP001. When GSAMPCB is specified, the IEFRDER batch log data set is not used by the catalog members information gather task. Only the catalog update task writes to the batch log data set for catalog database changes. GSAMPCB and DELETE are mutually exclusive.

LATEST
If an ACB already exists in the IMS system, do not save an instance of the ACB in an ACB library to the staging data set unless the instance in the ACB library has a later timestamp than the ACB in the IMS system.

This is the default.

UNCOND
If an ACB already exists in the IMS system, save an instance of the ACB in an ACB library to the staging data set unconditionally, unless the timestamp of the ACB in the ACB library is the same as the timestamp of the ACB in the IMS system.

gsamdbd

gsamdbd is the name of changed GSAM database. The specified database record is inserted into the IMS catalog, and the application control block is written to the staging or active directory based on the MANAGEDACBS control statement parameters that you specify.

You can use the *gsamdbd* variable with the STAGE or UPDATE parameter. However, LATEST, UNCOND, DELETE, SHARE, and GSAMPCB are not supported if you specify the *gsamdbd* variable.

Because the *gsamdbd* variable identifies a GSAM resource, the GSAMPCB batch logging rule applies only to the catalog update task that writes to the batch log data set for catalog database changes.

Follow these steps if you want to change GSAM DBDs by using the STAGE parameter and *gsamdbd*. This process writes only the GSAM DBD to the staging directory.

1. Change the GSAM DBD source.
2. Run a DBDGEN to generate the GSAM DBD into DBDLIB.
3. Run the DFS3PU00 utility with MANAGEDACBS=(STAGE,GSAM=*gsamdbd*).

   Tip: Only the DBDLIB DD card is required for the DFS3PU00 utility. ACBLIB and PSBLIB are not needed and are ignored if specified.

   The new GSAM DBD instance will be inserted to the catalog, and the application control block will be written to the staging directory.
4. Run the **IMPORT DEFN SOURCE(CATALOG)** or **IMPORT DEFN SOURCE(CATALOG) NAME(gsamdb)** command with or without OPTION(NOCHECK).

UPDATE

Updates existing IMS directory system data sets directly in exclusive mode. The ACBs are not placed in the staging data set.

Recommendation: Shut down all IMS systems that use the IMS directory when you specify the UPDATE parameter. When UPDATE is specified, the IMS Catalog Populate utility requires exclusive access to the IMS directory.

Also, the utility does not notify online IMS systems when it updates the IMS directory. Consequently, any IMS systems that are online when the utility updates the IMS directory must be restarted to load the updated ACBs.

When UPDATE is specified, the DFS3PU00 utility inserts ACBs that are not already in the IMS system into the IMS directory unconditionally. If an ACB exists in the IMS system, the instance of the ACB is replaced depending on the timestamp of the instance in the ACB library and the specification of the UNCOND or LATEST parameter.

Do not specify UPDATE when you do an initial load of the IMS catalog or before you enable the IMS management of ACBs. The UPDATE parameter is for updating existing IMS directory data sets only.

When UPDATE is specified, the IMS catalog PSB DFSCP001 must be specified in the utility JCL.

If you specify multiple parameters after MANAGEDACBS, you must enclose them in parentheses. For example, `(UPDATE,LATEST)`. If you specify only UPDATE, you can omit the parentheses.

By default, the system data set names that are allocated for IMS management of ACBs are derived from the data set name prefix that is specified in the RECON data set when the first partition of the IMS catalog is defined. To override this default, you can specify the **SYSDSHLQ=** parameter in a TYPE=CATDSHLQ statement in the dynamic allocation (DFSMDA) macro.

LATEST

If an ACB already exists in the IMS system, do not replace it with an instance of the ACB from an ACB library unless the instance in the ACB library has a later timestamp than the ACB in the IMS system.

This is the default.

UNCOND

If an ACB already exists in the IMS system, replace it with an instance of the ACB from an ACB library unconditionally, unless the timestamp of the ACB in the ACB library is the same as the timestamp of the ACB in the IMS system.

SHARE

For dynamic option (DOPT) PSBs only, allocates the required IMS directory data sets in a shared mode so that DOPT PSBs can be added to the IMS catalog without interrupting online processing.

Do not specify SHARE if the utility is loading the IMS catalog, as indicated by the specification of DFSCPL00, the IMS catalog load PSB, in the execution parameters of the utility JCL.

Do not specify SHARE if any resources in the input ACBLIB data sets are not DOPT PSBs.

Before you run the utility with the SHARE option specified, the extended sharing of PDSE data sets must be enabled in your z/OS system. The extended sharing of PDSE data sets is enabled in z/OS by specifying PDSESHARING(EXTENDED) in the IGDSMS*xx* member in the SYS1.PARMLIB on each system in the sysplex. If extended PDSE sharing is not enabled when SHARE is specified, the utility allocates the required IMS catalog data sets in exclusive mode, which might cause conflicts with other IMS processes and application programs.

For more information about enabling extended PDSE sharing, see z/OS: Specifying Extended PDSE Sharing in a Multiple-System Environment.

GSAMPCB
GSAM resources are included for MANAGEDACBS= running in DLI mode using PSB DFSCP001. When GSAMPCB is specified, the IEFRDER batch log data set is not used by the catalog members information gather task. Only the catalog update task writes to the batch log data set for catalog database changes. GSAMPCB and DELETE are mutually exclusive.

gsamdbd

gsamdbd is the name of changed GSAM database. The specified database record is inserted into the IMS catalog, and the application control block is written to the staging or active directory based on the MANAGEDACBS control statement parameters that you specify.

You can use the *gsamdbd* variable with the STAGE or UPDATE parameter. However, LATEST, UNCOND, DELETE, SHARE, and GSAMPCB are not supported if you specify the *gsamdbd* variable.

Because the *gsamdbd* variable identifies a GSAM resource, the GSAMPCB batch logging rule applies only to the catalog update task that writes to the batch log data set for catalog database changes.

Follow these steps if you want to change GSAM DBDs by using the UPDATE parameter and *gsamdbd*:

1. Change the GSAM DBD source.
2. Run a DBDGEN to generate the GSAM DBD into DBDLIB.
3. Run the DFS3PU00 utility with MANAGEDACBS=(UPDATE,GSAM=*gsamdbd*).

   Tip: Only the DBDLIB DD card is required for the DFS3PU00 utility. ACBLIB and PSBLIB are not needed and are ignored if specified.

   The new GSAM DBD instance will be inserted to the catalog, and the application control block will be written to the IMS directory.

NODUPLIST
Do not print a list of resource instances that were not added. This parameter is the default.

NOISRTLIST
Do not print a list of inserted resource instances.

Import statistics report

The IMS Catalog Import utility creates an import statistics report for the record segments to be loaded or updated in the IMS catalog. When the IMS Catalog Import utility runs in analysis-only mode, the report reflects only the potential statistics if the IMS catalog were loaded or updated from the ACB libraries currently being used as input to the utility.

The first section in the report is a summary of the segments that were inserted during the current execution of the utility. For each segment type inserted in the IMS catalog, the summary includes the following information organized into columns:

- The segment code (SC)
- The segment name, which generally corresponds to a macro in your DBD or PSB source files

- The data set group (DSG) in the IMS catalog that the segment is stored in
- The parent segment of the listed segment
- The total number of segments of that type that are loaded into the IMS catalog
- The average number of segment instances of that type under the parent segment

The second section in the report shows both the numbers of existing DBD and PSB records that were updated by the utility and the number of existing DBD and PSB records that were not updated because the timestamp of the record matched the timestamp of the corresponding ACB member in the ACB library. This section of the report also shows the number of DBD and PSB segments that do not need to be inserted because their timestamps in the IMS catalog match their timestamps in the ACB library.

The rest of the sections in the report provide the storage estimates.

For the OSAM data sets, the storage sections of the report show the number of blocks of the specified size. For the VSAM KSDSs, which include the indirect list data set (ILDS), the primary index data set, and the secondary index data set, the report shows the number of VSAM records.

These numbers are estimates that reflect the amount of space needed to load the catalog records that are built from the ACB libraries that you provide as input to the DFS3PU00 utility. If you are calculating the amount of storage required for the IMS catalog data sets, provide plenty of additional space in your calculations to allow for expansion.

If you have IMS create the IMS catalog data sets automatically, you can specify additional space as a percentage of the estimates that are provided by the utility on the SPACEALLOC parameter in the IMS catalog section of the DFSDFxxx PROCLIB member. The default value for this parameter is 500%.

In the report, the following abbreviations are used:

DSG
Data set group

L
A HALDB ILDS data set. The number of records shown represent the potential number of indirect list entries (ILEs) that could be created if the IMS catalog is reorganized.

SC
Segment code. When loading a segment type, IMS assigns a segment code as a unique identifier (an integer from 1 to 255). IMS assigns numbers in ascending sequence, starting with the root segment type (number 1) and continuing through all dependent segment types in hierarchical sequence.

SEGS
Segments

X
HALDB partitioned primary index.

```
CATALOG DFSCD000

PARTITION DFSCD01

   NUMBER OF SEGMENTS INSERTED INTO THE CATALOG
                  INSERTED                        AVERAGE
   SC   SEGMENT    SEGMENTS   DSG   PARENT      SEGS/PARENT
   --   --------   --------   ---   --------    -----------
    1   HEADER        4228     A
    2   DBD           2530     A    HEADER          0.6
    3   CAPXDBD          7     D    DBD             0.0
    5   DSET          2599     D    DBD             1.0
    7   AREA           139     D    DBD             0.1
    9   SEGM         16337     B    DBD             6.5
   10   CAPXSEGM         1     D    SEGM            0.0
   12   FLD          16426     C    SEGM            1.0
   14   MAR          16426     C    FLD             1.0
   17   LCHILD        2687     B    SEGM            0.2
   20   XDFLD          134     B    LCHILD          0.0
   33   PSB           1840     A    HEADER          0.4
   35   PCB           9190     B    PSB             5.0
   37   SS           75274     B    PCB             8.2
   39   SF            1105     B    SS              0.0
   41   DBDXREF       8886     D    PSB             4.8
```

```
            WITHIN    DUPLICATES
           EXISTING       NOT
SEGMENT     HEADER     INSERTED
-------   --------   ----------
DBD           71           0
PSB           72           0

ESTIMATED SPACE REQUIREMENT TO HOLD INSERTED SEGMENTS
  DSG   BLKSIZE   BLOCKS
  ---   -------   ------
   A      4096       596
   B      4096      9343
   C      4096      8214
   D      4096       236

  DSG   RECORDS
  ---   -------
   L      8886
   X      4230

  SECONDARY
  INDEX        RECORDS
  ---------    -------
  DFSCX000       8886
```

Library statistics report

The IMS Catalog Import utility calls the z/OS data set utility IEBCOPY to copy ACB, DBD, and PSB libraries. The IEBCOPY utility provides statistics and messages for the copy operation. For more information about the IEBCOPY utility, see the related link near the bottom of this topic.

Return codes

The IMS Catalog Import utility returns the following codes:

0
Successful completion of all operations

4
One or more warning messages were issued.

8 or greater
Program terminated due to severe errors

Chapter 9. IMS Catalog Directory Recovery utility (DFS3RU00)

Use the IMS™ catalog Directory Recovery utility (DFS3RU00) to rebuild the IMS directory data sets and recover the ACBs of IMS databases and programs to the directory by reading active resources from the IMS catalog, or clean up the IMS directory data sets. The utility is used in an IMS-managed ACB environment you are using DDL and cannot build IMS directory from an ACBLIB.

The DFS3RU00 utility has four basic functions:

- Rebuild IMS directory data sets
- Recover the ACBs for all IMS databases and programs
- Write online runtime blocks to the IMS directory data sets
- Clean up the IMS directory data sets

The functions of rebuilding, recovering, and writing are enabled by specify **MBR=ALL** in the **SYSIN DD** control statement.

The DFS3RU00 utility can also be submitted to clean up the IMS directory data sets when the directory online updates aborted abnormally.

In case of a disaster recovery scenario where both IMS catalog and the IMS directory data sets are lost, you must recover the IMS catalog database first before running this utility to rebuild and restore the directory data sets.

Subsections:

- "Restrictions" on page 353
- "Prerequisites" on page 353
- "Requirements" on page 353
- "Recommendations" on page 354
- "Input and Output" on page 354
- "JCL specifications" on page 354
- "Return codes" on page 355

Restrictions

The DFS3RU00 utility runs as DL/I batch region. The main storage databases (MSDB) and shared secondary index databases are not supported.

Prerequisites

In case of disaster recovery, you must recover IMS catalog first. IMS catalog acts as input to the DFS3RU00 utility.

Ensure that the DFS3RU00 utility does not clean up the directory data sets when either the **IMPORT DEFN SOURCE(CATALOG)** command or the DDL definition activation process is in progress of updating the directory data sets.

Requirements

The DFS3RU00 utility requires access to the DFSDF*xxx* member of the IMS.PROCLIB data set.

Before you run the DFS3RU00 utility, ensure that the IMS catalog DBDs (DFSCD000, DFSCX000) and PSBs (DFSCP000, DFSCP001) are recovered, so that the utility can be rerun in case of recovery failure. The DFS3RU00 utility writes the catalog DBDs and PSBs to the directory data sets. If the directory data

© Copyright IBM Corp. 1974, 2024

sets already exist, specify **ACBMGMT=CATALOG** in the DFSDF*xxx* member of the IMS.PROCLIB data set to enable the IMS catalog and the IMS management of ACBs. Otherwise, specify **ACBMGMT=ACBLIB**.

Recommendations

When you clean up the directory data sets with the DFS3RU00 utility, enable internal resource lock manage (IRLM) in the JCL and in the IMS system to avoid conflicts with the processes that update the directory data sets.

If IRLM is not used, issue the **QUERY MEMBER TYPE(IMS) SHOW()** command to check the status of the IMPORT and DDL definition activation processing.

Input and Output

The DFS3RU00 utility uses the following input:

- SYSIN DD statement
- CATALOG section of the DFSDF*xxx* member of the IMS.PROCLIB data set

The DFS3RU00 utility generates the following output:

- One or more IMS directory data sets that are recovered
- The bootstrap dataset

JCL specifications

EXEC statement

The following JCL shows the PARM parameters that you can use to run the DFS3RU00 utility. Use the **DFSDF=** parameter to specify the three-character suffix of your DFSDF*xxx* member in the IMS.PROCLIB data set.

```
//BATCH EXEC PGM=DFSRRC00,
// PARM=(DLI,DFS3RU00,DFSCP000,,,,,,,,,,Y,N,,,,,,,,,,,'DFSDF=xxx')
```

To enable IRLM support, you can specify the PARM parameters on the EXEC statement as shown in the following example:

```
// PARM=(DLI,DFS3RU00,DFSCP000,,,,,,,,,,Y,Y,irlmid,,,,,,,,,,,'DFSDF=xxx')
```

DD statements

STEPLIB DD
Points to the IMS.SDFSRESL library, which contains the IMS nucleus and the required IMS modules. If the **STEPLIB DD** statement is unauthorized because it contains unauthorized libraries that are concatenated to IMS.SDFSRESL, you must include a **DFSRESLB DD** statement.

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

For IMS batch, SDFSRESL and any data set that is concatenated to it on the **DFSRESLB DD** statement must be authorized by the Authorized Program Facility (APF). The **JOBLIB DD** and **STEPLIB DD** statements do not need to be authorized for IMS batch.

PROCLIB DD
Specifies the DFSDF*xxx* member of the IMS.PROCLIB data set that defines the default retention criteria for the records in the IMS catalog.

IMS DD
Defines the IMS.PSBLIB and IMS.DBDLIB data sets. The **IMS DD** statement is ignored if the IMS-management of ACBs is enabled.

IMSDBSDS DD
Optionally defines the IMS directory bootstrap data set (BSDS).

Use this DD statement to control the placement and size of the data set. If this statement is provided, the utility does not delete an existing BSDS and then create a new one, but overwrites the content of the existing data set.

The data set name must conform to the same naming convention as for a utility-created BSDS. The data set must be DSORG=PS, RECFM=FB, LRECL=96.

If this statement is omitted, the utility dynamically deletes any preexisting BSDS and dynamically creates a new one.

IMSDnnnn DD
Optionally defines the IMS directory data sets that are used to store the ACBs.

Use this DD statement for each IMS directory data set to control the number of data sets that are used, and the placement and size of each data set. You can provide this DD statement for up to 20 data sets. The number of DD statements that start from IMSD0001 determines how many data sets are used. The utility does not dynamically delete or create any of the directory data sets if IMSD0001 is specified.

You can specify only one data set on each DD statement. Concatenation is not allowed.

The data set name must conform to the same naming convention as for a system-created directory data set. Each data set must be a PDS-E with DSORG=PO, RECFM=U, LRECL=0, BLKSIZE=32760, DSNTYPE=LIBRARY. The data set must be empty when the utility begins processing the data set.

If the IMSD0001 DD is omitted, the utility dynamically deletes any preexisting directory data sets and dynamically creates two new data sets to store the ACBs.

IEFRDER DD
Defines the primary IMS log data set.

DFSVSAMP DD
Defines the buffer pool parameters data set.

SYSPRINT DD
Defines the output report data set.

SYSIN DD
Defines the input control statement. The utility accepts the following SYSIN DD control statement:

```
MBR=ALL
```

By specifying **MBR=ALL**, the utility reads the metadata of all active IMS resources from the IMS catalog, recovers the online runtime blocks, and writes them to the IMS directory data sets.

Return codes

The IMS Catalog Recovery utility completes execution with one of the following return codes:

0
No errors or exceptional conditions.

4
Errors in database type or IMS online block building failure. The DFS3RU00 utility does not support MSDB and shared secondary index databases. To restore the ACBs of these access type databases and programs that refer them or the ACBs for block builder failure resources, perform ACBGEN and then run the Catalog Populate utility (DFS3PU00) to recover these members to directory data sets.

8
Errors in control statements.

12
See description in the job report.

20
One or more supported object types were not recovered.

JCL examples

The following sample JCL recovers the ACBs for all databases and programs. Ellipses require replacement with user values.

```
//BATCH EXEC PGM=DFSRRC00,
// PARM=(DLI,DFS3RU00,DFSCP000,,,,,,,,,,Y,N,,,,,,,,,,,'DFSDF=001')
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//IMS      DD  DSN=IMS.PSBLIB,DISP=SHR
//         DD  DSN=IMS.DBDLIB,DISP=SHR
//IEFRDER  DD  ...
//DFSVSAMP DD  ...
//SYSPRINT DD  SYSOUT=*
//SYSIN DD *
 MBR=ALL
```

The following sample JCL cleans up directory data sets. The PSB name DFS3DCU0 must be specified on the **PARM=** parameter list. No control card is needed.

```
//BATCH EXEC PGM=DFSRRC00,
// PARM=(DLI,DFS3RU00,DFS3DCU0,,,,,,,,,,,Y,irlmid,,,,,,,,,,'DFSDF=CT2')
//STEPLIB DD DSN=IMSTESTL.TNUC0,DISP=SHR
// DD DSN=IMSTESTL.TNUCT,DISP=SHR
// DD DSN=IMSBLD.I14RTSMM.CRESLIB,DISP=SHR
// DD DSN=IMSTESTG.IMSTESTS.DYNALLOC,DISP=SHR
// DD DSN=IMSTESTG.IMSTESTL.DYNALLOC,DISP=SHR
//DFSRESLB DD DSN=IMSTESTL.TNUC0,DISP=SHR
// DD DSN=IMSBLD.I14RTSMM.CRESLIB,DISP=SHR
// DD DSN=IMSTESTG.IMSTESTS.DYNALLOC,DISP=SHR
// DD DSN=IMSTESTG.IMSTESTL.DYNALLOC,DISP=SHR
//PROCLIB DD DSN=IMSVS.PROCLIB2,DISP=SHR
// DD DSN=USER.PRIVATE.PROCLIB,DISP=SHR
// DD DSN=IMSTESTG.I14RTSMM.PROCLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//SYSOUT DD SYSOUT=A
//
```

Related concepts

Building the application control blocks (ACBGEN) (Database Administration)

Related tasks

Recovering the IMS catalog when IMS manages ACBs (Database Administration)

Cleaning up the IMS directory data sets (Database Administration)

Related reference

QUERY MEMBER command (Commands)

Chapter 10. IMS Catalog Library Builder utility (DFS3LU00)

When the IMS management of ACBs is enabled, you can use the IMS Catalog Library Builder utility (DFS3LU00) to create database descriptions (DBD), program specification blocks (PSB), and application control blocks (ACB) for use by tools or processes that still require them. The utility builds these resources by reading the IMS catalog.

The IMS Catalog Library Builder utility can also create DBD and PSB macro statements for input into the DBD Generation and PSB Generation utilities.

If DDL (database definition language) is being used, but other processes still require DBD, PSB, and ACB libraries, this utility can be used to update these libraries with the changes introduced by DDL.

Restrictions

No restrictions are documented for this utility.

Prerequisites

The IMS management of ACBs must be enabled.

Requirements

The output data sets must exist and contain enough free space to hold the members that are generated by the utility.

Input and output

The DFS3LU00 utility reads the IMS catalog to create one or more of the following resources from the definitions of the databases and program views that are stored in an IMS catalog:

- A library of the ACBs that are currently active in the IMS system.
- A library of DBDs for the databases that are currently active in the IMS system.
- A library of PSBs for the program views that are currently active in the IMS system.
- DBD macro statement definitions of the databases that are currently active in the IMS system. The macro definitions can be used as input to the DBD Generation utility.
- PSB macro statement definitions of the program views that are currently active in the IMS system. The macro definitions can be used as input to the PSB Generation utility.

The resources that the utility can create are selected by specifying control statement parameters on the SYSIN DD statement.

For each control statement that is provided as input, the utility requires an output data set that is identified by a DD statement of the same name as the control statement.

© Copyright IBM Corp. 1974, 2024

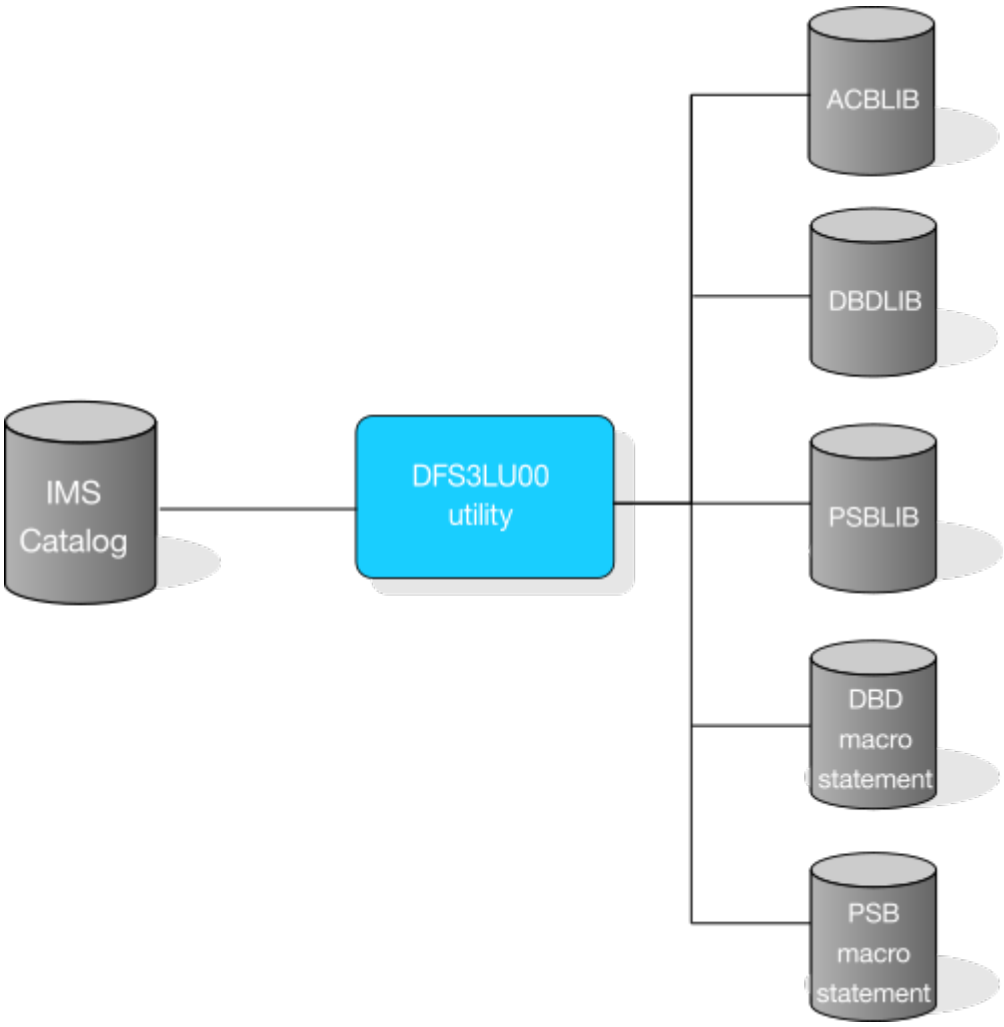


Figure 33. IMS Catalog Library Builder utility input and output

JCL specifications

The IMS Catalog Library Builder utility can generate DBD macro statements, PSB macro statements, and ACB, DBD, and PSB libraries. Depending on the control statements and DD statements that you include in the JCL, the utility can generate one, some, or all of these different types of output in a single execution.

The DD statements that you include must not contain concatenations.

In the JCL, the IMSCATHLQ parameter identifies the IMS catalog that contains the database and program view definitions that the utility uses to build the DBD and PSB resources. You can specify either the high-level qualifier of the IMS directory data sets of the IMS catalog or the name of a CATDSHLQ DFSMDA member that defines the high-level qualifier of the IMS directory data sets. For an example of the JCL that specifies a CATDSHLQ DFSMDA member, see JCL for specifying a CATDSHLQ DFSMDA member.

JCL for creating DBD macro statements

```
//*******************************************************************
//* THIS STEP FILLS THE DBDSOR LIBRARY
//*******************************************************************
//LUDBDSOR EXEC PGM=DFS3LU00,REGION=0M
//*
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//LUSYSPRT DD  SYSOUT=A
//SYSOUT   DD  SYSOUT=A
//SYSABEND DD  SYSOUT=*
//DBDSOR   DD  DSN=IMS.DFS3LU00.DBDSOR,DISP=OLD,DCB=(RECFM=FB,LRECL=80)
```

```
//SYSIN    DD  *
 DBDSOR
 IMSCATHLQ=IMS.DFSCD000
/*
```

JCL for building PSB macro statements

```
//*******************************************************************
//* THIS STEP FILLS THE PSBSOR LIBRARY
//*******************************************************************
//LUPSBSOR EXEC PGM=DFS3LU00,REGION=0M
//*
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//LUSYSPRT DD  SYSOUT=A
//SYSOUT   DD  SYSOUT=A
//SYSABEND DD  SYSOUT=*
//PSBSOR   DD  DSN=IMS.DFS3LU00.PSBSOR,DISP=OLD,DCB=(RECFM=FB,LRECL=80)
//SYSIN    DD  *
 PSBSOR
 IMSCATHLQ=IMS.DFSCD000
/*
```

JCL for building an ACB library

```
//*******************************************************************
//* THIS STEP FILLS THE ACBLIB LIBRARY
//*******************************************************************
//LUACBLIB EXEC PGM=DFS3LU00,REGION=0M
//*
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//LUSYSPRT DD  SYSOUT=A
//SYSOUT   DD  SYSOUT=A
//SYSABEND DD  SYSOUT=*
//ACBLIB   DD  DSN=IMS.DFS3LU00.ACBLIB,DISP=OLD
//SYSIN    DD  *
 ACBLIB
 IMSCATHLQ=IMS.DFSCD000
/*
```

JCL for building a PSB library

```
//*******************************************************************
//* THIS STEP FILLS THE PSBLIB LIBRARY
//*******************************************************************
//LUPSBLIB EXEC PGM=DFS3LU00,REGION=0M
//*
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
//SYSLIB   DD  DISP=SHR,DSN=IMS.SDFSMAC
//SYSPRINT DD  SYSOUT=A
//LUSYSPRT DD  SYSOUT=A
//SYSOUT   DD  SYSOUT=A
//SYSABEND DD  SYSOUT=*
//PSBLIB   DD  DSN=IMS.DFS3LU00.PSBLIB,DISP=OLD
//SYSIN    DD  *
 PSBLIB
 IMSCATHLQ=IMS.DFSCD000
/*
//*******************************************************************
//*  SYSAIN is used to compile the PSB source.
//*******************************************************************
//SYSAIN   DD  DSN=IMSTESTS.DFS3LU00.SYSAIN,DISP=OLD
//*******************************************************************
//*  SYSLIN is used to link the compiled PSB objects into libraries.
//*******************************************************************
//SYSLIN   DD  DSN=IMSTESTS.DFS3LU00.SYSLIN,DISP=OLD
//*
```

JCL for building a DBD library

```
//*******************************************************************
//* THIS STEP FILLS THE DBDLIB LIBRARY
//*******************************************************************
//LUDBDLIB EXEC PGM=DFS3LU00,REGION=0M
//*
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
//SYSLIB   DD  DISP=SHR,DSN=IMS.SDFSMAC
//SYSPRINT DD  SYSOUT=A
//LUSYSPRT DD  SYSOUT=A
//SYSOUT   DD  SYSOUT=A
//SYSABEND DD  SYSOUT=*
//DBDLIB   DD  DSN=IMS.DFS3LU00.DBDLIB,DISP=OLD
//SYSIN    DD  *
 DBDLIB
 IMSCATHLQ=IMS.DFSCD000
/*
//*******************************************************************
//*  SYSAIN is used to compile the DBD source.
//*******************************************************************
//SYSAIN   DD  DSN=IMSTESTS.DFS3LU00.SYSAIN,DISP=OLD
//*******************************************************************
//*  SYSLIN is used to link the compiled DBD objects into libraries.
//*******************************************************************
//SYSLIN   DD  DSN=IMSTESTS.DFS3LU00.SYSLIN,DISP=OLD
//*
```

JCL for specifying a CATDSHLQ DFSMDA member

In the following example, the data set that contains the DFSMDA member is concatenated to the STEPLIB. The alias of the IMS catalog and the characters HLQ are specified on the IMSCATHLQ parameter. This value matches the name of DFSMDA member that defines the high-level qualifier.

This method for specifying a CATDSHQ DFSMDA member in the DFS3LU00 utility JCL for building any type of resource.

```
//*******************************************************************
//* THIS STEP FILLS THE ACBLIB LIBRARY
//*******************************************************************
//LUACBLIB EXEC PGM=DFS3LU00,REGION=0M
//*
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//         DD  DSN=IMS.MDALIB,DISP=SHR
//DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//LUSYSPRT DD  SYSOUT=A
//SYSOUT   DD  SYSOUT=A
//SYSABEND DD  SYSOUT=*
//ACBLIB   DD  DSN=IMS.DFS3LU00.ACBLIB,DISP=OLD
//SYSIN    DD  *
 ACBLIB
 IMSCATHLQ=DFSCHLQ
/*
```

DD statements

ACBLIB DD
Defines the output IMS ACBLIB data sets to contain members that the utility creates.

DBDLIB DD
Defines the output IMS DBDLIB data set to contains the members that the utility creates.

DBDSOR DD
Defines the output data set to contain the DBD source members that the utility creates.

You must specify the following DCB attributes with the DBDSOR DD statement:

- Organization . . . . : PO
- Record format . . . : F or FB

- Record length . . . : 80
- Block size . . . . . : 80 or a multiple of 80 if RECFM=FB
- Data set name type : PDS

DFSRESLB DD
Points to an authorized library that contains the IMS SVC modules. For IMS batch, SDFSRESL and any data set that is concatenated to it on the DFSRESLB DD statement must be authorized through the Authorized Program Facility (APF). This DD statement provides an authorized library for the IMS SVC modules, which must be in an authorized library. The JOBLIB or STEPLIB statement does not need to be authorized for IMS batch.

LUSYSPRT DD
Defines the output data set that holds status and summary statistical information that is written by the utility. You can send this data set to SYSOUT or define it as a permanent data set.

The following example shows the types of status messages that can be written to the LUSYSPRT DD statement for each member processed. The output of the compiler and linker contains additional information.

```
COMPILE SUCCESS FOR MEMBER : mbr_name
LINK    SUCCESS FOR MEMBER : mbr_name
COMPILE FAILURE FOR MEMBER : mbr_name  RC = rc
LINK    FAILURE FOR MEMBER : mbr_name  RC = rc
```

If the utility runs to completion, the utility also writes summary statistics to the LUSYSPRT DD statement. The following example shows the types of statistics that are written by the utility:

```
Processed xxxxxxxx out of yyyyyyyy members
Total catalog entries      : xxxxxxxx
Entry get failures        : xxxxxxxxx
Entry get successes       : xxxxxxxxx
xxxxxxxx compiles attempted : yyyyyyyy
xxxxxxxx compiles completed : yyyyyyyy
xxxxxxxx compiles failed    : yyyyyyyy
xxxxxxxx links attempted    : yyyyyyyy
xxxxxxxx links successful   : yyyyyyyy
xxxxxxxx links failed       : yyyyyyyy
xxxxxxxx members created    : yyyyyyyy
```

PROCLIB DD
Defines the IMS.PROCLIB data set that contains the DFSDFxxx member that defines various attributes of the IMS catalog that are required by the utility.

PSBLIB DD
Defines the output IMS PSBLIB data set to contain the members that the utility creates.

PSBSOR DD
Defines the output data set to contain the PSB source members that the utility creates.

You must specify the following DCB attributes with the PSBSOR DD statement::

- Organization . . . . : PO
- Record format . . . : F or FB
- Record length . . . : 80
- Block size . . . . . : 80 or a multiple of 80 if RECFM=FB
- Data set name type : PDS

STEPLIB DD
Points to IMS.SDFSRESL, which contains the IMS nucleus and required IMS modules. If STEPLIB is unauthorized by having unauthorized libraries that are concatenated to IMS.SDFSRESL, you must include a DFSRESLB DD statement.

SYSABEND DD
Defines the dump data set.

SYSAIN DD
When the DBDLIB or PSBLIB control statement is specified, defines an intermediate data set that is used as input to the compiler that creates the DBDLIB or PSBLIB members. Define the data set with `LRECL=80,BLKSIZE=80,RECFM=F,DSORG=PS` or `LRECL=80, BLKSIZE=nn (a multiple of 80), RECFM=FB, DSORG=PS`.

SYSIN DD
Defines the input control statement data sets. They can be on a tape volume, direct-access device, card reader, or be routed through the input stream. The input can be blocked as multiples of 80. During execution, this utility can process as many control statements as required.

SYSLIB DD

When the DBDLIB or PSBLIB control statement is specified, identifies the IMS.SDFSMAC data set that contains the IMS macro libraries.

SYSLIN DD
When the DBDLIB or PSBLIB control statement is specified, defines an intermediate data set that is used as input to the binder that links the DBDLIB or PSBLIB members. Define the data set with `LRECL=80,BLKSIZE=80,RECFM=F,DSORG=PS` or `LRECL=80, BLKSIZE=nn (a multiple of 80),RECFM=FB, DSORG=PS`.

SYSPRINT DD
Defines the output message data set.

When the SYSPRINT DD statement refers to a DASD or tape data set, you can control the block size of this data set with the BLKSIZE subparameter of the **DCB** parameter. If specified, the BLKSIZE value must be an exact multiple of 121 or a system ABEND013-20 can result. Omitting BLKSIZE from a DASD data set causes a system-determined block size to be used. Regardless of what value is specified for the LRECL parameter, the utility always uses a record length of 121.

Utility control statements

You can specify any combination of the following control parameters on the SYSIN DD statement for the DFS3LU00 utility.

ACBLIB
Copies the ACBs that are active in the IMS system to an ACB library data set. The resulting ACB library can be used by IMS systems, application programs, utilities, or other processes that require an ACBLIB data set.

IMSCATHLQ=
A required control statement that specifies the high-level qualifier of the IMS directory data sets from which the utility retrieves the ACB, DBD, and PSB resources.

If the high-level qualifier of the IMS directory data sets is defined by a CATDSHLQ DFSMDA member, specify the alias name of the IMS catalog followed by HLQ. For example, `IMSCATHLQ=DFSCHLQ`. This value must also match the value that is specified on the DDNAME parameter of the CATDSHLQ DFSMDA member that defines the high-level qualifier.

DBDLIB
Creates database descriptor (DBD) members in a DBD library data set from the active database definitions that are stored in the IMS catalog.

When the DBDLIB control statement is specified, the following additional DD statements are required:

- SYSAIN DD
- SYSLIB DD
- SYSLIN DD

MBR=
Specifies the name of a single resource for which the utility output is to be created. The specified name must match the name of an active resource in the IMS catalog.

The type of resource that is specified on the MBR= parameter must be consistent with the output type that is defined by the control statement that is specified with the MBR= parameter. For example, if the DBDLIB or DBDSOR control statement is specified, the resource that is named on the MBR= parameter must be a DBD.

The MBR= parameter can be specified only once per utility execution. If it is specified multiple times, the utility creates only the resource that is specified on the last instance of the MBR= parameter.

PSBLIB
Creates program specification block (PSB) members in a PSB library data set from the active program view (PSB) definitions that are stored in the IMS catalog.

When the PSBLIB control statement is specified, the following DD statements are also required:

- SYSAIN DD
- SYSLIB DD
- SYSLIN DD

DBDSOR
Creates macro statements for the DBD Generation utility from the active database definitions that are stored in the IMS catalog.

PSBSOR
Creates macro statements for the PSB Generation utility from the active program view (PSB) definitions that are stored in the IMS catalog.

Return codes

The IMS Catalog Library Builder utility returns the following codes:

0
There was a successful completion of all operations.

4
One or more warning messages were issued.

8
The program terminated due to severe errors.

12 or greater
The program terminated due to severe errors.

Chapter 11. IMS Catalog Maintenance utility (DFS3CM00)

Use the IMS Catalog Maintenance utility (DFS3CM00) to fix incorrect entities in the catalog such as header timestamps and PARTYPE values.

The DFS3CM00 utility can be run in either a DL/I region or a BMP region. In the following example, the second Y and the *irlmnm* value indicate IRLM support:

```
PARM=(DLI,DFS3CM00,DFSCP001,,,,,,,,,,Y,Y,irlmnm,Y,,,,,,,,,'DFSDF=001')
```

Note: The information used by this utility to perform fixes in the catalog is pulled from the directory, and the utility does not fix anything in the directory, only in the catalog.

Subsections:

- “Restrictions” on page 365
- “Requirements” on page 365
- “Input and output” on page 365
- “JCL specifications” on page 365
- “Control statements” on page 366
- “Return codes” on page 367

Restrictions

This utility cannot fix the timestamps in the catalog header when run in a non-MACB environment.

Requirements

This utility requires access to both the catalog and the directory. The utility must be run with DFSCP001 PSB specified in the JCL.

Input and output

This utility always reads input from the SYSIN DD card in the JCL.

The output of the DFS3CM00 utility includes:

- A message for each object whose timestamp is fixed.
- A summary of total timestamps fixed.
- Error and warning messages for issues such as invalid input, missing instances, and missing objects.

JCL specifications

The execution parameters for the DFS3CM00 utility must reference the DFSDFxxx member of the IMS.PROCLIB data set, as shown in the following example JCL.

```
//DFS3CM00 EXEC PGM=DFSRRC00,
//             PARM=(DLI,DFS3CM00,DFSCP001,,,,,,,,,,Y,Y,irlmnm,Y,,,,,,,,,
//             ,'DFSDF=001')
//PROCLIB  DD DISP=SHR,
//      DSN=USER.PRIVATE.AUTOSRVR.PROCLIB
//STEPLIB     DD  DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
```

© Copyright IBM Corp. 1974, 2024

```
//SYSIN    DD *
TIMESTAMP
/*
//DFSVSAMP
DD

//SYSPRINT DD
SYSOUT=*
//SYSABEND DD SYSOUT=*
//IEFRDER  DD
```

The DFSCP001 PSB is used to obtain update authority. The procedure should also include STEPLIB and DFSRESLB DD statements for the IMS.SDFSRESL data set and IEFRDER and IEFRDER2 DD statements for the IMS log data sets. The IMS DD statement is not required if IMS-management of ACBs is enabled. The backout parameter BKO= should also be set when running the utility in a DLI region.

The utility can be run as BMP by specifying BMP and the IMS id in the PARM list, as shown in the following example.

```
//DFS3CM00 EXEC PGM=DFSRRC00,
//              PARM=(BMP,DFS3CM00,DFSCP001,,,,,,,,,,imsid)
```

DD statements

DFSRESLB DD
Points to an authorized library that contains the IMS SVC modules. For IMS batch, SDFSRESL and any data set that is concatenated to it on the DFSRESLB DD statement must be authorized through the Authorized Program Facility (APF). This DD statement provides an authorized library for the IMS SVC modules, which must be in an authorized library. The JOBLIB or STEPLIB statement does not need to be authorized for IMS batch.

DFSVSAMP
Defines the buffer pool parameters data set.

IEFRDER DD
Defines the primary IMS log data set.

IEFRDER2 DD
Defines the secondary IMS log data set.

PROCLIB DD
Defines the IMS.PROCLIB data set that contains the DFSDFxxx member that defines various attributes of the IMS catalog that are required by the utility.

IMS DD
Defines the IMS.PSBLIB and IMS.DBDLIB data sets.

The IMS DD statement is not required if IMS management of ACBs is enabled.

STEPLIB DD
Points to IMS.SDFSRESL, which contains the IMS nucleus and required IMS modules. If STEPLIB is unauthorized by having unauthorized libraries that are concatenated to IMS.SDFSRESL, you must include a DFSRESLB DD statement.

SYSABEND DD
Defines the dump data set.

SYSPRINT DD
Defines the output message data set.

Control statements

TIMESTAMP
Reads and fixes the catalog header timestamps, if any are determined to be incorrect or inconsistent with those present in the directory.

PARTYPE
Fixes incorrect PARTYPE values.

ALL
Fixes both incorrect timestamps and incorrect PARTYPE values.

No input or blank input
Fixes both incorrect timestamps and incorrect PARTYPE values. This is the default.

RESOURCE_CHKP_FREQ=n
Specifies the number of objectsresources to process between checkpoints.

n can be a 1- to 8-digit numeric value ranging from 1 to 99999999. The default value is 100.

Return codes

The DFS3CM00 utility returns the following codes:

0
All operations completed successfully.

4
While fixing timestamps, the Catalog Maintenance utility could not find the expected directory instance in the catalog.

8
An expected object was not found in the directory. The active or pending timestamp of an object in the catalog was non-zero and not found in the corresponding active or staging directory.

12
Invalid input provided in SYSIN; unable to read the directory.

36
The Catalog Maintenance utility was unable to retrieve the HLQ for the catalog directory datasets. The output will also include a DFS9113E message. This can happen if the utility was run in a non-MACB environment.

Related reference
DFSDFxxx member of the IMS PROCLIB data set (System Definition)
“Application Control Blocks Maintenance utility (DFSUACB0)” on page 3
In IMS systems that use ACB libraries to manage runtime application control blocks, use the Application Control Blocks (ACB) Maintenance utility to save instruction execution and direct-access wait time and improve performance in application scheduling.

Chapter 12. IMS Catalog Partition Definition Data Set utility (DFS3UCD0)

The IMS Catalog Partition Definition Data Set utility (DFS3UCD0) creates and populates the IMS catalog partition definition data set. This data set stores information about IMS catalog database partitions when DBRC is not used to manage the database partitions.

Restrictions

No restrictions are documented for this utility.

Prerequisites

No prerequisites are documented for this utility.

Requirements

The utility creates the specified partition definition data set if it is empty. If the data set already contains data, the existing information is overwritten.

Recommendations

Use DBRC to manage the IMS catalog partition information when possible.

Input and output

The DFS3UCD0 utility uses the following input and output data sets:

- The DFSHDBSC data set contains the input and output from the utility.
- The IMS DBDLIB data set is used as an input data set for the utility.
- The SYSPRINT data set that receives messages from the utility.
- The SYSIN data set that contains the utility control statements.

JCL specifications

EXEC statement

```
//S1       EXEC PGM=DF3UCD0
```

DD statements

JOBLIB/STEPLIB
The location of the IMS.SDFSRESL data set, which contains the executable modules for this utility.

DFSHDBSC
The data set name that is used for the input to and output from the utility. Only one data set name is allowed and concatenated data sets are not valid. The DCB values for this data set are RECFM=F and LRECL=80.

IMS
The IMS DBDLIB data set.

SYSPRINT
The data set that receives messages generated by the utility. The DCB parameters for the **SYSPRINT** data set are RECFM=FB and LCRECL=133.

© Copyright IBM Corp. 1974, 2024

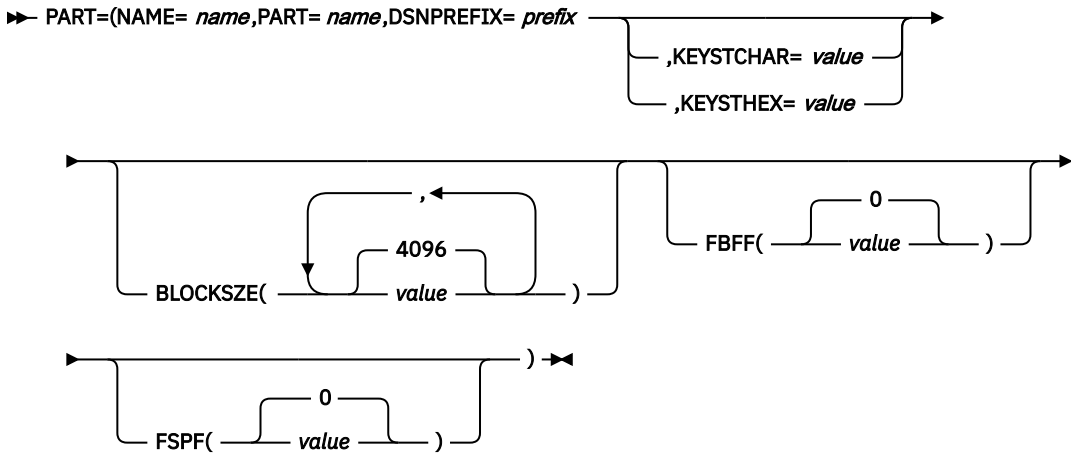
SYSIN
The data set that contains the input parameters for the utility. The DCB parameters for the **SYSIN** data set are RECFM=FB and LRECL=80.

Only submit **HALDB** and **PART** statements with the **SYSIN** DD statement for this utility.

HALDB statement
This statement specifies the HALDB master name of the IMS catalog.

NAME
Specifies the name of the IMS catalog HALDB that you are defining in the IMS catalog partition definition data set.

PART statement
This statement specifies the structure of the partition or partitions in the IMS catalog HALDB.



NAME
Identifies the name of the HALDB that this partition is being defined for.

PART
Specifies a HALDB partition name. This value can be up to 7 alphanumeric characters long. The first character must be alphabetic.

DSNPREFIX
Specifies a data set name prefix for the partition data sets contained in the partition. This value can be up to 37 characters and must be a valid JCL data set name.

KEYSTCHAR
This value is the HALDB partition high key value for this partition. The high key cannot be longer than the root key. If the high key is shorter than the root key, the high key value is padded with repeating X'FF' bytes up to the defined root key length. Each partition high key value must be unique within a HALDB. This value can be up to 256 characters long.

Either this parameter or **KEYSTHEX** is required. This parameter is mutually exclusive with **KEYSTHEX**.

KEYSTHEX
Specifies a HALDB partition high key value in hexadecimal format. It can be up to 512 characters long. Either this parameter or **KEYSTCHAR** is required. This parameter is mutually exclusive with **KEYSTCHAR**.

BLOCKSZE
Specifies the block size for OSAM data sets. This value must be an even number up to 32766. This value is used for OSAM data sets only. The default value for this parameter is 4096. You can specify up to 10 values for this parameter, one for each data set group defined in the DBD. Separate multiple values with commas.

FBFF
Specifies that every nth control interval or block in this data set group is left as free space during database load or reorganization operations. This value can be any whole number between 0 and 100 except 1. The default value is 0, which specifies that no free space is retained during load or reorganization operations.

FSPF
Specifies the minimum percentage of free space that must be retained in this data set group. This value can be any number between 0 and 99. The default is 0.

Return codes

This utility generates a DFS4353I message after it completes. The message contains one of the following return codes:

0
Processing completed successfully.

40
The BPESTART macro did not complete successfully. The DFS4353I reason code field contains the return code from the BPESTART macro.

44
Unable to obtain storage. The DFS4353I reason code indicates what storage could not be obtained:

1
BPEPARSE grammar

2
Parse output block

3
HALDB definition descriptor block

4
DBD record storage

48
An error occurred while parsing the utility SYSIN statement. The DFS4353I reason code field contains the return code from the BPEPARSE macro.

52
An error occurred while validating the utility SYSIN statement.

56
An error occurred while reading the utility SYSIN statement using the BPERDPDS macro. The DFS4353I reason code field contains the return code from the BPERDPDS macro.

60
An error occurred while opening SYSPRINT.

76
The HALDB partition definition data set (DFSHDBSC) could not be processed. The reason code indicates when the processing error occurred:

72
(RC_OPENDEFDS) - An error occurred while opening the data set.

80
(RC_CLOSEDEFDS) - An error occurred while closing the data set.

84
(RC_RDJCBDEFDS) - An RDJFCB macro error occurred while processing the data set.

88
The DBDLIB could not be processed. The reason code indicates the underlying cause:

1
No IMS DD statement was found.

2
An IMS DD open error occurred.

3
The specified database was not found in the DBDLIB.

4
The specified database is not a HALDB.

Sample JCL statements

```
//S1       EXEC PGM=DFS3UCD0,REGION=0M
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//DFSHDBSC  DD  DSN=...,DISP=
//SYSPRINT DD  SYSOUT=*
//IMS      DD  DSN=IMS.DBDLIB,DISP=SHR
//SYSIN    DD  *
HALDB=(NAME=DFSCD000)
PART=(NAME=DFSCD000,PART=xxxxxxxx,
      DSNPREFX=xxxxxxxx,
      KEYSTCHAR=xxxxxxxx)
/*
```

Chapter 13. IMS Catalog Populate utility (DFS3PU00)

Use the IMS Catalog Populate utility (DFS3PU00) to load, insert, or update DBD and PSB instances into the database data sets of the IMS catalog from ACB library data sets. If the IMS management of ACBs is enabled in the IMS system, the utility also adds the active application control blocks (ACBs) to the IMS directory data set that is managed by IMS.

You can also use the DFS3PU00 utility to estimate the required sizes of the IMS catalog data sets without updating the IMS catalog. The utility generates the size estimates by scanning the members in the input ACB libraries.

Whether the DFS3PU00 utility runs in load mode, update mode, or analysis-only mode is determined by the IMS catalog PSB that you specify in the utility JCL:

- Load mode is specified by PSB DFSCPL00.
- Update mode is specified by PSB DFSCP001.
- Analysis-only mode is specified by DFSCP000.

If the IMS management of ACBs is enabled, you must specify the MANAGEDACBS= control statement on the SYSINP DD statement to make the generated ACBs available to the online IMS system. Depending on the specification, the DFS3PU00 utility either adds the ACBs to the staging data set of the IMS catalog for later activation or, if the IMS system is offline or you are setting up managed ACBs for the first time, adds them directly to the IMS directory data sets. IMS loads the ACBs from the directory data sets during startup.

If dynamic option (DOPT) PSBs are the only resource type in the input ACB libraries, the utility can add the DOPT PSBs without taking the system offline. Some restrictions apply. See the SHARE keyword under “Control statements” on page 381.

If any of the data sets for the IMS catalog have not been created, including those for the IMS directory, the DFS3PU00 utility creates them automatically. The IMS directory, an internal IMS structure, is used to hold the runtime control blocks for the DBD and PSB instances that are flagged as active in the IMS catalog.

The amount of space that the utility allocates for the IMS catalog data sets is based on the members in the input ACB libraries and the values that are specified on SPACEALLOC parameter in the catalog section of the DFSDFxxx PROCLIB member. The amount of space that the utility allocates for the IMS directory data sets, which are PDSE data sets, is also based on the size of the ACB library, but is increased dynamically as needed.

The output of the DFS3PU00 utility includes a report that contains statistics about the record segments that are loaded into the IMS catalog. The report includes information about the number and types of segments, as well as an estimate of the amount of DASD storage that each data set group of the IMS catalog will require. If you need to know how much DASD storage the IMS catalog data sets will use before they are created, you can run the DFS3PU00 utility without populating the IMS catalog to generate only the statistics report. To run the DFS3PU00 utility in analysis-only mode, specify DFSCP000 as the PSB for the utility in the utility JCL.

After the utility evaluates the members in your ACB libraries, it produces a report. This report is the same report that the DFS3PU00 utility produces when it loads the IMS catalog.

The DFS3PU00 utility creates the catalog records from the ACB members in one or more ACB libraries and depending on your database types, the associated DBD and PSB members in DBD libraries and PSB libraries. The records contain metadata for your application programs and databases.

The DFS3PU00 utility can run in a DL/I region, or if it is updating an existing IMS catalog, the DFS3PU00 utility can run in a BMP region. If you run the utility in a DL/I batch region and the IMS catalog is shared,

© Copyright IBM Corp. 1974, 2024

you must specify IRLM support in the EXEC parameters of the utility JCL. In the following example, the second Y and *irlmid* value indicate IRLM support:

```
PARM=(DLI,DFS3PU00,DFSCP001,,,,,,,,,,Y,Y,irlmid,,,,,,,,,,'DFSDF=001')
```

As an alternative to running the DFS3PU00 utility, you can populate the IMS catalog by using the ACB Generation and Catalog Populate utility (DFS3UACB). The DFS3UACB utility generates the ACB libraries for your applications and databases and then populates the IMS catalog, both in the same job step.

Subsections:

- "Restrictions" on page 374
- "Prerequisites" on page 374
- "Requirements" on page 374
- "Recommendations" on page 375
- "Input and output" on page 375
- "JCL specifications" on page 376
- "Control statements" on page 381
- "IMS Catalog Populate utility statistics report" on page 385
- "Return codes" on page 387

Restrictions

The IMS Catalog Populate utility (DFS3PU00) cannot be run as a BMP in an ACBMGMT=ACBLIB environment when the MANAGEDACBS=SETUP utility control statement is specified.

Prerequisites

Before the DFS3PU00 utility can load the metadata for new or changed application programs and databases into the IMS catalog, the DBD generation, PSB generation, and ACB generation processes must be complete for the new or changed application programs and databases.

If you are loading an IMS catalog for the first time, ensure that the following steps have been completed before running the DFS3PU00 utility:

- The DBD and PSB load modules for the IMS catalog have been added to your DBD and PSB libraries.
- The ACB member for the IMS catalog has been generated and loaded into the IMS.ACBLIB data set.
- The IMS catalog HALDB master database and partitions have been defined in either the RECON data set or, if the target IMS catalog is not supported by DBRC, in an IMS catalog partition definition data set.

Before you can use the SHARE option to add DOPT PSBs to an online IMS catalog, extended sharing of PDSE data sets must be enabled in z/OS. For more information, see z/OS: Specifying Extended PDSE Sharing in a Multiple-System Environment.

Requirements

The DFS3PU00 utility requires access to the following data sets:

- The IMS.PROCLIB data set that contains the DFSDFxxx member that enables the IMS catalog and defines the alias name of the IMS catalog
- One or more IMS.ACBLIB data sets
- If ACB library members reference logically related databases, the IMS.DBDLIB data set
- If ACB library members reference GSAM databases, the IMS.DBDLIB data set and the IMS.PSBLIB data set

When the IMS.DBDLIB and IMS.PSBLIB data sets are included as input, they must be the DBD and PSB libraries from which the input IMS.ACBLIB data set was built.

If any required members of the PSBLIB or DBDLIB data sets are not found, the utility issues an error message, and the records for the referencing PSBs are not created in the IMS catalog. You can add the missing catalog records later by supplying the necessary PSB or DBD members with the correct ACB library and rerunning the DFS3PU00 utility in update mode by specifying the DFSCP001 PCB in the utility JCL.

If the IMS management of ACBs is enabled, you must specify the MANAGEDACBS= control statement on the SYSINP DD statement to make the generated ACBs available to the online IMS system.

When the IMS catalog is registered with DBRC, you are required to create an image copy of the IMS catalog after an initial load of the IMS catalog. When the IMS catalog is not registered with DBRC, IMS cannot require an image copy after an initial load; however, if an image copy is not created after an initial load, the only way to recover the catalog is to reload it.

Recommendations

If you are updating an existing IMS catalog, create an image copy of the IMS catalog data sets when the updates are complete. If the IMS catalog is registered with DBRC, you can use the DBRC command **GENJCL.IC** to back up the catalog. If you have defined the IMS catalog in an IMS Catalog partition definition data set, you must use standard image copy JCL.

The storage requirement is highly dependent on the total number of ACBLIB members because they have to be kept in storage. Consider increasing the job's region size to prevent a GETMAIN failure that will cause abend U1002 reason code 4 to be issued.

Input and output

The DFS3PU00 utility always reads input from the ACB library data sets that contain the ACB members for your databases and application programs, and from the DFSDFxxx member of the IMS.PROCLIB data set.

If your databases use logical relationships, the IMS Catalog Populate utility also reads input from the IMS.DBDLIB data set.

If you use GSAM databases and an ACB library member references a GSAM database, the DFS3PU00 utility also reads input from both the IMS.DBDLIB and the IMS.PSBLIB data sets.

The output of the DFS3PU00 utility includes:

- The record segments in the IMS catalog data set (DFSCD000).
- Messages and statistical information to the SYSPRINT data set.
- If MANAGEDACBS=SETUP is specified:
  - New or updated record segments in the IMS catalog
  - IMS directory data sets
  - ACBs in the IMS directory data sets that correspond to the active DBD and PSB instances in the IMS catalog
- If MANAGEDACBS=STAGE is specified:
  - New or updated record segments in the IMS catalog
  - ACBs in the staging data set of the IMS directory data sets
- If MANAGEDACBS=UPDATE is specified:
  - New or updated record segments in the IMS catalog
  - ACBs in the IMS directory data sets that correspond to the active DBD and PSB instances in the IMS catalog

If any of the following database data sets do not exist, the DFS3PU00 utility creates them automatically:

- The DFSCD000 database data sets:
  - Four data sets for the segments of the IMS catalog
  - The indirect list data set (ILDS)
  - The primary index data set
- The DFSCX000 secondary index data set.
- When the MANAGEDACBS=SETUP statement is specified, the IMS directory data sets, including the IMS directory bootstrap data set.

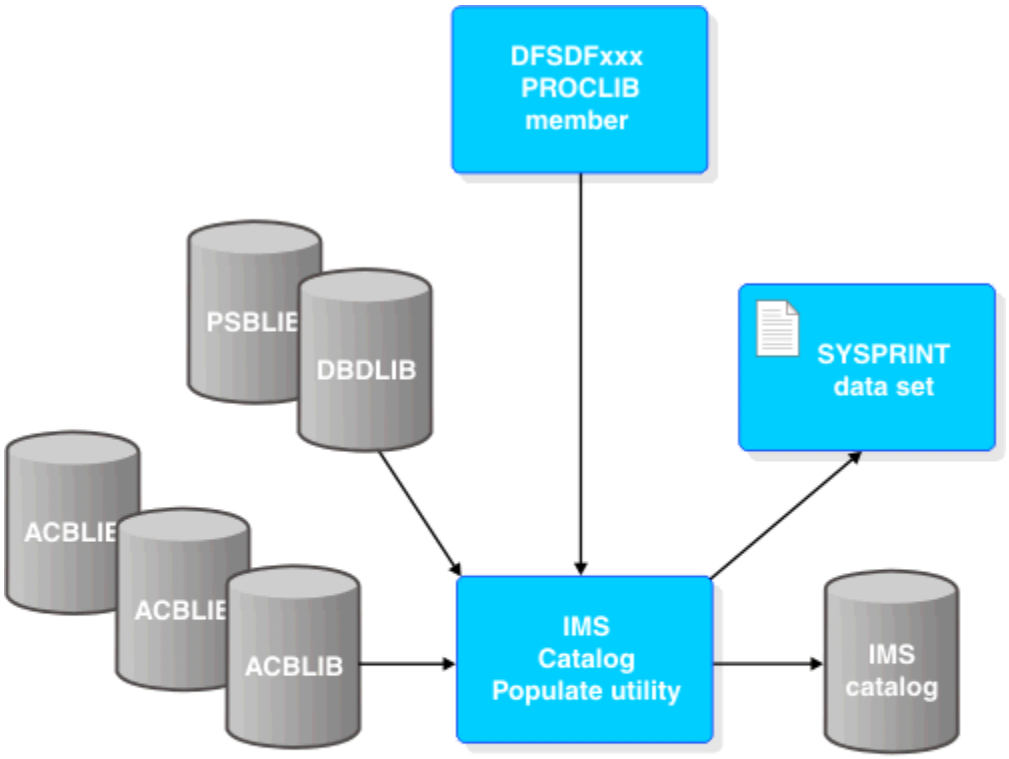


Figure 34. IMS Catalog Populate utility input and output

JCL specifications

DFS3PU00 utility JCL statements

The execution parameters for the DFS3PU00 utility must reference the DFSDFxxx member of the IMS.PROCLIB data set, as shown in the following example JCL.

At least one IMS.ACBLIB data set must be specified in the utility JCL by the IMSACB01 DD statement. You can include additional ACBLIB data sets by concatenating them in a single DD statement or by specifying additional IMSACB*nn* DD statements, as shown in the following example JCL by the IMSACB02 and IMSACB03 DD statements. The additional IMSACB*nn* ddnames must be consecutively numbered in the last two characters positions.

Duplicate ACB members with duplicate names are handled differently depending on whether the ACB libraries are concatenated on a single DD statement or are referenced individually by separate DD statements. If you use cloned ACB libraries, but duplicate ACB members might be generated individually in any of the libraries, reference each of the cloned ACB libraries in the utility JCL by using a separate DD statement.

When ACB libraries are referenced individually by separate DD statements, the DFS3PU00 utility checks the ACB generation time stamps of ACB members with duplicate names and uses them only if the ACB generation time stamp is different from the previously processed ACB member with the same name. If the time stamps are the same, the ACB member with the duplicate name is ignored.

However, when ACB libraries are concatenated in a single DD statement, the DFS3PU00 utility does not check the ACB generation time stamps of ACB members with duplicate names. If duplicate member names exist across the concatenated ACB libraries, only the first ACB member is used and any subsequent ACB members with a duplicate name are ignored, even if the time stamps are different.

Your DLIBATCH or equivalent procedure should already include IMS DD statements for the PSB and DBD libraries that were used to generate the ACB libraries. The procedure should also include STEPLIB and DFSRESLB DD statements for the IMS.SDFSRESL data set and IEFRDER and IEFRDER2 DD statements for the IMS log data sets.

The following is a sample of the JCL statements that can be used to perform an initial load of the IMS catalog with the DFS3PU00 utility. The JCL specifies the IMS catalog PSB DFSCPL00, which loads records to the IMS catalog. Any existing records are overwritten.

```
//LOADCAT EXEC PGM=DFS3PU00,
// PARM=(DLI,DFS3PU00,DFSCPL00,,,,,,,,,,Y,N,,,,,,,,,,,'DFSDF=001')
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
//IMS      DD  DSN=IMS.PSBLIB,DISP=SHR
//         DD  DSN=IMS.DBDLIB,DISP=SHR
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//SYSABEND DD  SYSOUT=*       Dump data set
//SYSPRINT DD  SYSOUT=*       Messages, statistics
//IEFRDER  DD  ...            Log data set
//DFSVSAMP DD  ...            Buffer pool parameters
//IMSACB01 DD  ...            First ACBLIB
//         DD  ...            Optional concatenated ACBLIB
//IMSACB02 DD  ...            Optional additional ACBLIBs
//IMSACB03 DD  ...            ...
//SYSINP   DD  *              ISRTLIST DUPLIST /*
```

The following is a sample of the JCL statements that can be used to update the IMS catalog by running the DFS3PU00 utility in a DL/I batch job. The JCL specifies the IMS catalog PSB DFSCP001, which inserts records to the IMS catalog without replacing the entire catalog. IRLM support is not indicated.

```
//UPDTCAT EXEC PGM=DFS3PU00,
//            PARM=(DLI,DFS3PU00,DFSCP001,,,,,,,,,,Y,N,,,,,,,,,
//            ,,'DFSDF=001')
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
//IMS      DD  DSN=IMS.PSBLIB,DISP=SHR
//         DD  DSN=IMS.DBDLIB,DISP=SHR
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//SYSABEND DD  SYSOUT=*       Dump data set
//SYSPRINT DD  SYSOUT=*       Messages, statistics
//IEFRDER  DD  ...            Log data set with catalog updates
//DFSVSAMP DD  ...            Buffer pool parameters
//IMSACB01 DD  ...            First ACBLIB
//         DD  ...            Optional concatenated ACBLIB
//IMSACB02 DD  ...            Optional additional ACBLIBs
//IMSACB03 DD  ...            ...
```

The following is a sample of the JCL statements that can be used to update the IMS catalog by running the DFS3PU00 utility in a DL/I batch job. The JCL specifies the IMS catalog PSB DFSCP001, which inserts records to the IMS catalog without replacing the entire catalog. IRLM support is indicated and is necessary if authorization of the catalog database exists.

```
//UPDTCAT EXEC PGM=DFS3PU00,
//     PARM=(DLI,DFS3PU00,DFSCP001,,,,,,,,,,Y,Y,irlmid,,,,,,,,,
//             ,,'DFSDF=001')
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
//IMS      DD  DSN=IMS.PSBLIB,DISP=SHR
//         DD  DSN=IMS.DBDLIB,DISP=SHR
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//SYSABEND DD  SYSOUT=*       DUMP DATA SET
//SYSPRINT DD  SYSOUT=*       MESSAGES, STATISTICS
//IEFRDER  DD  ...            LOG DATA SET WITH CATALOG UPDATES
//DFSVSAMP DD  ...            BUFFER POOL PARAMETERS
//IMSACB01 DD  ...            FIRST ACBLIB
//         DD  ...            OPTIONAL CONCATENATED ACBLIB
//IMSACB02 DD  ...            OPTIONAL ADDITIONAL ACBLIBS
```

The following is a sample of the JCL statements that can be used to update the IMS catalog by running the DFS3PU00 utility in a BMP job. The JCL specifies the IMS catalog PSB DFSCP001, which inserts records to the IMS catalog without replacing the entire catalog.

In this example, *imsid* is the identifier of the IMS system on which the job is to be run.

```
//UPDTCAT EXEC PGM=DFS3PU00,
  //             PARM=(BMP,DFS3PU00,DFSCP001,,,,,,,,,,imsid,,,,,,)
  //STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
  //DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
  //IMS      DD  DSN=IMS.PSBLIB,DISP=SHR
  //         DD  DSN=IMS.DBDLIB,DISP=SHR
  //PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
  //SYSABEND DD  SYSOUT=*       Dump data set
  //SYSPRINT DD  SYSOUT=*       Messages, statistics
  //IEFRDER  DD  ...            Log data set with catalog updates
  //DFSVSAMP DD  ...            Buffer pool parameters
  //IMSACB01 DD  ...            First ACBLIB
  //         DD  ...            Optional concatenated ACBLIB
  //IMSACB02 DD  ...            Optional additional ACBLIBs
  //IMSACB03 DD  ...            ...
```

DD statements

DFSRESLB DD

Points to an authorized library that contains the IMS SVC modules. For IMS batch, SDFSRESL and any data set that is concatenated to it on the DFSRESLB DD statement must be authorized through the Authorized Program Facility (APF). This DD statement provides an authorized library for the IMS SVC

modules, which must be in an authorized library. The JOBLIB or STEPLIB statement does not need to be authorized for IMS batch.

DFSVSAMP
Defines the buffer pool parameters data set.

IEFRDER DD
Defines the primary IMS log data set.

IEFRDER2 DD
Defines the secondary IMS log data set.

IMS DD
Defines the IMS.PSBLIB and IMS.DBDLIB data sets.

The IMS DD statement is required only if the IMS ACB library members loaded into the IMS catalog reference GSAM or logically related databases.

IMSACB01 DD
Defines an ACB library data set that contains the ACB members that are used to populate the IMS catalog. This DD statement is required.

IMSACB*nn* DD
Defines additional optional input ACB library data sets. The ddnames of additional ACB libraries must be consecutively numbered in the *nn* position.

The value of *nn* is determined by adding 1 to the value of *nn* from the preceding IMSACB*nn* ddname. Valid values for *nn* are from 02 through 99.

For example, after the required IMSACB01 DD statement, subsequent DD statements would be named IMSACB02, IMSACB03, IMSACB04, and so on.

If a break in the consecutive numbering of IMSACB*nn* ddnames occurs, the ACB library DD statements after the break are ignored.

IMSDBSDS DD
Optionally defines the IMS directory bootstrap data set (BSDS).

Use this DD statement to control the placement and size of the data set. If this statement is provided, the utility does not delete an existing BSDS and then create a new one, but overwrites the content of the existing data set.

The data set name must conform to the same naming convention as for a utility-created BSDS. The data set must be DSORG=PS, RECFM=FB, LRECL=96.

If this statement is omitted, the utility dynamically deletes any preexisting BSDS and dynamically creates a new one.

IMSDnnnn DD
Optionally defines the IMS directory data sets that are used to store the ACBs.

Use this DD statement for each IMS directory data set to control the number of data sets that are used, and the placement and size of each data set. You can provide this DD statement for up to 20 data sets. The number of DD statements that start from IMSD0001 determines how many data sets are used. The utility does not dynamically delete or create any of the directory data sets if IMSD0001 is specified.

You can specify only one data set on each DD statement. Concatenation is not allowed.

The data set name must conform to the same naming convention as for a system-created directory data set. Each data set must be a PDS-E with DSORG=PO, RECFM=U, LRECL=0, BLKSIZE=32760, DSNTYPE=LIBRARY. The data set must be empty when the utility begins processing the data set.

If the IMSD0001 DD is omitted, the utility dynamically deletes any preexisting directory data sets and dynamically creates two new data sets to store the ACBs.

IMSDG001 DD

An optional control statement to define an empty work data set to be used as an IMS.ACBLIB data set for the IMS Catalog Populate utility (DFS3PU00). This data set is used to temporarily hold ACB library members that reference a GSAM database. The DFS3PU00 utility then moves the ACB library members into the IMS directory data sets after it copies the ACB library members from the libraries specified on the IMSACB*nn* DD statement.

Provide this DD to obtain more control over the placement and size of the temporary data set. If this DD is provided the DFS3PU00 utility does not dynamically create or delete a temporary data set.

This data set has the following attributes:

DSORG
Partitioned data set extended (PDSE)

DSNTYPE
LIBRARY

RECFM
U

LRECL
80

BLKSIZE
32760

If the IMS management of ACBs is not enabled, this statement is omitted.

If the data set is not a PDSE data set, the DFS4549E message is displayed.

If the data set is not empty when the DFS3PU00 utility begins processing, the content of the data set is deleted.

If this DD statement is omitted, the DFS3PU00 utility dynamically creates a temporary data set and deletes it after the utility completes processing.

IMSDSTAG DD
Optionally defines the IMS directory staging data set.

Provide this DD statement to control the placement and size of the data set. If this DD statement is provided, the utility does not dynamically delete or create the staging data set.

When you name the data set, conform to the same naming convention as for the staging data set. The data set must be a PDS-E with DSORG=PO, RECFM=U, LRECL=0, BLKSIZE=32760, DSNTYPE=LIBRARY.

The data set must be empty when the utility begins processing the data set.

If this DD statement is omitted, the utility dynamically deletes any preexisting staging data set and dynamically creates a new one.

PROCLIB DD
Defines the IMS.PROCLIB data set that contains the DFSDFxxx member that defines various attributes of the IMS catalog that are required by the utility.

STEPLIB DD
Points to IMS.SDFSRESL, which contains the IMS nucleus and required IMS modules. If STEPLIB is unauthorized by having unauthorized libraries that are concatenated to IMS.SDFSRESL, you must include a DFSRESLB DD statement.

SYSABEND DD
Defines the dump data set.

SYSINP DD
An optional control statement sequential data set with 80-character fixed-length records. Only characters in positions 1 - 72 are read.

The control statement parameters, which are separated by blanks or commas, can be specified on one or more records.

For a description of the control statements that you can specify with the SYSINP DD statement, see "Control statements" on page 381.

SYSPRINT DD
Defines the output message data set.

When the SYSPRINT DD statement refers to a DASD or tape data set, you can control the block size of this data set with the BLKSIZE sub-parameter of the **DCB** parameter. If specified, the BLKSIZE value must be an exact multiple of 121 or a system ABEND013-20 can result. Omitting BLKSIZE from a DASD data set causes a system-determined block size to be used. Regardless of what value is specified for LRECL, the utility will always use a record length of 121.

Control statements

The following control statement parameters, which are separated by blanks or commas, can be specified on one or more records by using the SYSINP DD statement.

DUPLIST
Specifies that the DFS3PU00 utility list each DBD or PSB resource in the input ACB library that is not added to the IMS catalog because it is a duplicate of an instance that is already in the IMS catalog. For each duplicate instance of a resource in the IMS catalog, the utility prints message DFS4436I.

If the MANAGEDACBS= statement is specified and the IMS management of ACBs is enabled, the utility also lists each DBD or PSB resources that is not added to the IMS directory or the staging data set because it is a duplicate of the instance of a resource that is already in the IMS directory.

If the UPDATE parameter is specified on the MANAGEDACBS= statement, the utility prints message DFS4531I for each duplicate instance.

If LATEST is specified or accepted as the default with the UPDATE parameter, the utility also prints message DFS4522I for each instance that is not added to the IMS directory because the instance of the resource in the IMS directory has a later timestamp.

If the STAGE parameter is specified on the MANAGEDACBS= statement, the utility prints message DFS4542I for each resource that is not copied to the staging data set because it is a duplicate instance.

If LATEST is specified or accepted as the default with the STAGE parameter, the utility also prints message DFS4539I for each instance that is not added to the staging data set because the instance of the resource in the IMS directory has a later timestamp.

ERRORMAX=*n*
Terminate the IMS Catalog Populate utility when more than *n* messages indicate errors that prevent certain DBDs and PSBs from having their metadata that is written to the IMS catalog. Duplicate instances of metadata do not count toward this limit. If this option is omitted, there is no limit.

RESOURCE_CHKP_FREQ=*n*
Specifies the number of DBD and PSB resource instances to be inserted between checkpoints. *n* can be a 1- to 8-digit numeric value of 1 to 99999999. The default value is 100.

SEGMENT_CHKP_FREQ=*n*
Specifies the number of segments to be inserted between checkpoints. When the number is reached, IMS finishes inserting all of the segments for the resource instance that is currently being processed before issuing the checkpoint. *n* can be a 1- to 8-digit numeric value of 1 to 99999999. The default value is 1000.

Note: The first checkpoint frequency number to be reached will cause a checkpoint to be taken and the counters will be reset to 0.

ISRTLIST

If the IMS management of ACBs is enabled, the utility also lists each DBD or PSB resources that is either added to the IMS directory or saved to the staging data set for importing into the IMS directory later.

The utility identifies the action taken for each resource by printing the following messages:

DFS4520I
The resource was added to the IMS directory as a new resource.

DFS4521I
This instance of the resource replaced an existing instance of the resource in the IMS directory.

DFS4537I
The resource was copied to the staging data set. When it is imported into the IMS directory later, it will be inserted as a new resource in the IMS directory.

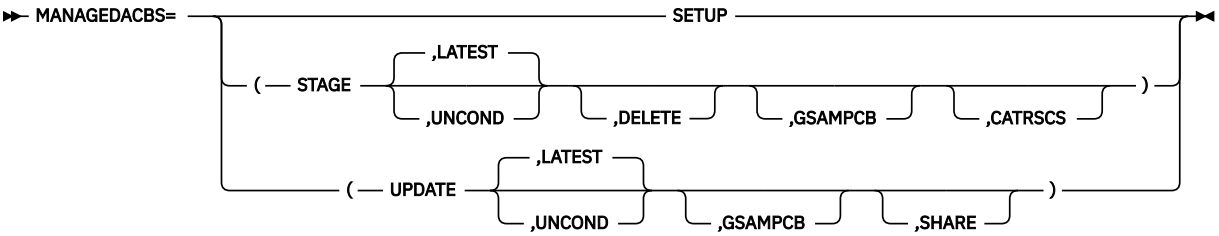
DFS4538I
The resource was copied to the staging data set. When it is imported into the IMS directory later, it will replace an existing instance of the resource in the IMS directory.

MANAGEDACBS=

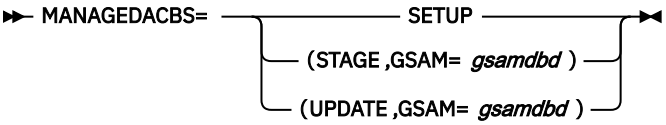
Use the MANAGEDACBS control statement to perform the following actions:

- Set up IMS to manage the runtime application control blocks (ACBs) for your databases and program views.
- Update an IMS system that manages ACBs with new or modified ACBs from an ACB library data set.
- Save ACBs from an ACB library to a staging data set for later importing into an IMS system that manages ACBs.

The MANAGEDACBS statement can be specified according to the following syntax diagram:



If you use the *gsamdbd* parameter, the MANAGEDACBS statement can be specified according to the following syntax diagram:



The parentheses are required only if you specify multiple parameters on the MANAGEDACBS= statement.

The following list describes the parameters that you can specify on the MANAGEDACBS= statement:

SETUP
Creates the IMS directory data sets that are required by IMS to manage application control blocks (ACBs). Any existing instances of the IMS directory data sets are replaced.

Do not specify ACBMGMT=CATALOG in the CATALOG section of the DFSDFxxx PROCLIB member until after you successfully run the utility with MANAGEDACBS=SETUP specified. You must create the IMS directory and load it with the active ACBs before you enable the IMS management of ACBs.

When SETUP is specified, the utility inserts the ACBs that are in the input ACB libraries into the IMS directory data sets.

If the IMS catalog PSB DFSCP001 is specified in the utility JCL, the utility inserts any new or modified DBDs or PSBs into the IMS catalog.

If the IMS catalog PSB DFSCPL00 is specified in the utility JCL, the utility deletes all existing catalog records and reloads the IMS catalog.

When the utility adds an ACB to the IMS directory, it flags the corresponding DBD and PSB instances in the IMS catalog as active.

Do not run the DFS3PU00 utility as a BMP with **MANAGEDACBS=SETUP** specified.

STAGE

Saves the ACBs from the input ACB libraries to a staging data set. ACBs saved in the staging data set are not activated until you add them to the IMS system by issuing the IMPORT DEFN SOURCE(CATALOG) command.

When STAGE is specified, the IMS catalog PSB DFSCP001 must be specified in the utility JCL.

If the staging data set exists when the MANAGEDACBS=STAGE control statement is specified, the utility uses the existing staging data set. Any ACBs already in the staging data set are preserved. To scratch and re-create the staging data set when you stage new resources, specify MANAGEDACBS=(STAGE,DELETE). When DELETE is specified, any ACBs in the existing data set are lost.

When STAGE is specified, the DFS3PU00 utility copies all ACBs in the ACB libraries that do not already exist in the IMS system into the staging data set. If an ACB in the ACB library already exists in the IMS system, the utility saves it to the staging data set based on whether timestamp of the ACB meets the criteria of the UNCOND or LATEST parameter.

Do not specify STAGE when you do an initial load of the IMS catalog or before you enable the IMS management of ACBs. The STAGE parameter is for staging updates to existing IMS directory data sets only.

If you specify multiple parameters after MANAGEDACBS=, you must enclose them in parentheses. For example, (STAGE,LATEST). If you specify only STAGE, you can omit the parentheses.

CATRSCS
When you use the STAGE option with the CATRSCS option, IMS catalog resources are added to the staging directory data set.

DELETE
If the staging data set is not allocated to any online IMS system, scratch and recreate the staging data set before adding the resources to the staging data set. Any ACBs in the existing staging data set are lost.

GSAMPCB
GSAM resources are included for MANAGEDACBS= running in DLI mode using PSB DFSCP001. When GSAMPCB is specified, the IEFRDER batch log data set is not used by the catalog members information gather task. Only the catalog update task writes to the batch log data set for catalog database changes. GSAMPCB and DELETE are mutually exclusive.

LATEST
If an ACB already exists in the IMS system, do not save an instance of the ACB in an ACB library to the staging data set unless the instance in the ACB library has a later timestamp than the ACB in the IMS system.

This is the default.

UNCOND
If an ACB already exists in the IMS system, save an instance of the ACB in an ACB library to the staging data set unconditionally, unless the timestamp of the ACB in the ACB library is the same as the timestamp of the ACB in the IMS system.

gsamdbd

gsamdbd is the name of changed GSAM database. The specified database record is inserted into the IMS catalog, and the application control block is written to the staging or active directory based on the MANAGEDACBS control statement parameters that you specify.

You can use the *gsamdbd* variable with the STAGE or UPDATE parameter. However, LATEST, UNCOND, DELETE, SHARE, and GSAMPCB are not supported if you specify the *gsamdbd* variable.

Because the *gsamdbd* variable identifies a GSAM resource, the GSAMPCB batch logging rule applies only to the catalog update task that writes to the batch log data set for catalog database changes.

Follow these steps if you want to change GSAM DBDs by using the STAGE parameter and *gsamdbd*. This process writes only the GSAM DBD to the staging directory.

1. Change the GSAM DBD source.
2. Run a DBDGEN to generate the GSAM DBD into DBDLIB.
3. Run the DFS3PU00 utility with MANAGEDACBS=(STAGE,GSAM=*gsamdbd*).

   Tip: Only the DBDLIB DD card is required for the DFS3PU00 utility. ACBLIB and PSBLIB are not needed and are ignored if specified.

   The new GSAM DBD instance will be inserted to the catalog, and the application control block will be written to the staging directory.
4. Run the **IMPORT DEFN SOURCE(CATALOG)** or **IMPORT DEFN SOURCE(CATALOG) NAME(gsamdb)** command with or without OPTION(NOCHECK).

UPDATE

Updates existing IMS directory system data sets directly in exclusive mode. The ACBs are not placed in the staging data set.

Recommendation: Shut down all IMS systems that use the IMS directory when you specify the UPDATE parameter. When UPDATE is specified, the IMS Catalog Populate utility requires exclusive access to the IMS directory.

Also, the utility does not notify online IMS systems when it updates the IMS directory. Consequently, any IMS systems that are online when the utility updates the IMS directory must be restarted to load the updated ACBs.

When UPDATE is specified, the DFS3PU00 utility inserts ACBs that are not already in the IMS system into the IMS directory unconditionally. If an ACB exists in the IMS system, the instance of the ACB is replaced depending on the timestamp of the instance in the ACB library and the specification of the UNCOND or LATEST parameter.

Do not specify UPDATE when you do an initial load of the IMS catalog or before you enable the IMS management of ACBs. The UPDATE parameter is for updating existing IMS directory data sets only.

When UPDATE is specified, the IMS catalog PSB DFSCP001 must be specified in the utility JCL.

If you specify multiple parameters after MANAGEDACBS, you must enclose them in parentheses. For example, (UPDATE,LATEST). If you specify only UPDATE, you can omit the parentheses.

By default, the system data set names that are allocated for IMS management of ACBs are derived from the data set name prefix that is specified in the RECON data set when the first partition of the IMS catalog is defined. To override this default, you can specify the **SYSDSHLQ=** parameter in a TYPE=CATDSHLQ statement in the dynamic allocation (DFSMDA) macro.

LATEST

If an ACB already exists in the IMS system, do not replace it with an instance of the ACB from an ACB library unless the instance in the ACB library has a later timestamp than the ACB in the IMS system.

This is the default.

UNCOND
If an ACB already exists in the IMS system, replace it with an instance of the ACB from an ACB library unconditionally, unless the timestamp of the ACB in the ACB library is the same as the timestamp of the ACB in the IMS system.

SHARE
For dynamic option (DOPT) PSBs only, allocates the required IMS directory data sets in a shared mode so that DOPT PSBs can be added to the IMS catalog without interrupting online processing.

Do not specify SHARE if the utility is loading the IMS catalog, as indicated by the specification of DFSCPL00, the IMS catalog load PSB, in the execution parameters of the utility JCL.

Do not specify SHARE if any resources in the input ACBLIB data sets are not DOPT PSBs.

Before you run the utility with the SHARE option specified, the extended sharing of PDSE data sets must be enabled in your z/OS system. The extended sharing of PDSE data sets is enabled in z/OS by specifying PDSESHARING(EXTENDED) in the IGDSMS*xx* member in the SYS1.PARMLIB on each system in the sysplex. If extended PDSE sharing is not enabled when SHARE is specified, the utility allocates the required IMS catalog data sets in exclusive mode, which might cause conflicts with other IMS processes and application programs.

For more information about enabling extended PDSE sharing, see z/OS: Specifying Extended PDSE Sharing in a Multiple-System Environment.

GSAMPCB
GSAM resources are included for MANAGEDACBS= running in DLI mode using PSB DFSCP001. When GSAMPCB is specified, the IEFRDER batch log data set is not used by the catalog members information gather task. Only the catalog update task writes to the batch log data set for catalog database changes. GSAMPCB and DELETE are mutually exclusive.

gsamdbd

gsamdbd is the name of changed GSAM database. The specified database record is inserted into the IMS catalog, and the application control block is written to the staging or active directory based on the MANAGEDACBS control statement parameters that you specify.

You can use the *gsamdbd* variable with the STAGE or UPDATE parameter. However, LATEST, UNCOND, DELETE, SHARE, and GSAMPCB are not supported if you specify the *gsamdbd* variable.

Because the *gsamdbd* variable identifies a GSAM resource, the GSAMPCB batch logging rule applies only to the catalog update task that writes to the batch log data set for catalog database changes.

Follow these steps if you want to change GSAM DBDs by using the UPDATE parameter and *gsamdbd*:

1. Change the GSAM DBD source.
2. Run a DBDGEN to generate the GSAM DBD into DBDLIB.
3. Run the DFS3PU00 utility with MANAGEDACBS=(UPDATE,GSAM=*gsamdbd*).

   Tip: Only the DBDLIB DD card is required for the DFS3PU00 utility. ACBLIB and PSBLIB are not needed and are ignored if specified.

   The new GSAM DBD instance will be inserted to the catalog, and the application control block will be written to the IMS directory.

NODUPLIST
Do not print a list of resource instances that were not added. This parameter is the default.

NOISRTLIST
Do not print a list of inserted resource instances.

IMS Catalog Populate utility statistics report

Each time the DFS3PU00 utility runs it creates a statistics report for the record segments to be loaded or updated in the IMS catalog. When the DFS3PU00 utility runs in read-only mode, the report reflects only the potential statistics if the IMS catalog were loaded or updated from the ACB libraries currently being used as input to the utility.

To run the DFS3PU00 utility in analysis-only mode, specify DFSCP000 as the PSB for the utility in the utility JCL.

The first section in the report is a summary of the segments that were inserted during the current execution of the utility. For each segment type inserted in the IMS catalog, the summary includes the following information organized into columns:

- The segment code (SC)
- The segment name, which generally corresponds to a macro in your DBD or PSB source files
- The data set group (DSG) in the IMS catalog that the segment is stored in
- The parent segment of the listed segment
- The total number of segments of that type that are loaded into the IMS catalog
- The average number of segment instances of that type under the parent segment

The second section in the report shows both the numbers of existing DBD and PSB records that were updated by the utility and the number of existing DBD and PSB records that were not updated because the time stamp of the record matched the time stamp of the corresponding ACB member in the ACB library. This section of the report also shows the number of DBD and PSB segments whose ACBLIB time stamps were already in the catalog and did not need to be inserted again.

The rest of the sections in the report provide the storage estimates.

For the OSAM data sets, the storage sections of the report show the number of blocks of the specified size. For the VSAM KSDSs, which include the indirect list data set (ILDS), the primary index data set, and the secondary index data set, the report shows the number of VSAM records.

These numbers are estimates that reflect the amount of space needed to load the catalog records that are built from the ACB libraries that you provide as input to the DFS3PU00 utility. If you are calculating the amount of storage required for the IMS catalog data sets, provide plenty of additional space in your calculations to allow for expansion.

If you have IMS create the IMS catalog data sets automatically, you can specify additional space as a percentage of the estimates that are provided by the utility on the SPACEALLOC parameter in the IMS catalog section of the DFSDFxxx PROCLIB member. The default value for this parameter is 500%.

In the report, the following abbreviations are used:

DSG
Data set group

L
A HALDB ILDS data set. The number of records shown represent the potential number of indirect list entries (ILEs) that could be created if the IMS catalog is reorganized.

SC
Segment code. When loading a segment type, IMS assigns a segment code as a unique identifier (an integer from 1 to 255). IMS assigns numbers in ascending sequence, starting with the root segment type (number 1) and continuing through all dependent segment types in hierarchical sequence.

SEGS
Segments

X
HALDB partitioned primary index.

```
 CATALOG DFSCD000

 PARTITION DFSCD01

    NUMBER OF SEGMENTS INSERTED INTO THE CATALOG
```

```
                INSERTED                        AVERAGE
SC   SEGMENT    SEGMENTS   DSG   PARENT     SEGS/PARENT
--   --------   --------   ---   --------   -----------
 1   HEADER        4228     A
 2   DBD           2530     A    HEADER         0.6
 3   CAPXDBD          7     D    DBD            0.0
 5   DSET          2599     D    DBD            1.0
 7   AREA           139     D    DBD            0.1
 9   SEGM         16337     B    DBD            6.5
10   CAPXSEGM         1     D    SEGM           0.0
12   FLD          16426     C    SEGM           1.0
14   MAR          16426     C    FLD            1.0
17   LCHILD        2687     B    SEGM           0.2
20   XDFLD          134     B    LCHILD         0.0
33   PSB           1840     A    HEADER         0.4
35   PCB           9190     B    PSB            5.0
37   SS           75274     B    PCB            8.2
39   SF            1105     B    SS             0.0
41   DBDXREF       8886     D    PSB            4.8

           WITHIN    DUPLICATES
          EXISTING       NOT
SEGMENT    HEADER     INSERTED
-------   --------   ----------
DBD           71           0
PSB           72           0

ESTIMATED SPACE REQUIREMENT TO HOLD INSERTED SEGMENTS
  DSG   BLKSIZE   BLOCKS
  ---   -------   ------
   A      4096       596
   B      4096      9343
   C      4096      8214
   D      4096       236

  DSG   RECORDS
  ---   -------
   L      8886
   X      4230

  SECONDARY
  INDEX        RECORDS
  ---------    -------
  DFSCX000       8886
```

Return codes

The DFS3PU00 utility returns the following codes:

0
All operations completed successfully.

4
One or more records could not be loaded into the IMS catalog. Warning messages issued.

8
One or more records could not be loaded into the IMS catalog. Error messages issued.

Greater than 8
Program terminated due to severe errors.

Related tasks

Populating the IMS catalog using the IMS Catalog Populate utility (DFS3PU00) (System Definition)

Related reference

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

“Application Control Blocks Maintenance utility (DFSUACB0)” on page 3

In IMS systems that use ACB libraries to manage runtime application control blocks, use the Application Control Blocks (ACB) Maintenance utility to save instruction execution and direct-access wait time and improve performance in application scheduling.

Chapter 14. IMS Catalog Record Purge utility (DFS3PU10)

Use the IMS Catalog Record Purge utility (DFS3PU10) to remove the segments that represent a DBD or PSB instance, all instances of a DBD version, or an entire DBD or PSB record from the IMS catalog.

If ACBMGMT=CATALOG is specified in the DFSDFxxx member of the PROCLIB data set, IMS also automatically removes from the IMS directory the DBDs and PSBs that are removed from the IMS catalog. If ACBMGMT=CATALOG is not specified in the DFSDFxxx member, use the MANAGEDACBS statement to enable the utility to remove from the IMS directory the DBDs and PSBs that are removed from the IMS catalog. When ACBs are removed from the IMS directory, the ACBs are also removed from the IMS directory staging data set.

The utility performs three basic functions, analysis, purge, and update, which can be run independently or sequentially in a single execution of the utility. Use the MODE control statement to select the analysis function, the purge function, or both. Use the UPDATE control statement to set or modify the retention criteria of DBD and PSB records in the IMS catalog.

The analysis function of the utility evaluates the records in the IMS catalog by reading the user-defined retention criteria to identify DBD and PSB instances that are eligible for deletion. When the DELDBVER statement is specified, the utility identifies all instances of the specified DBD version as eligible for deletion. The utility creates a report and DELETE statements for each DBD or PSB instance that can be deleted.

The purge function of the utility deletes the DBD and PSB instances by processing the DELETE statements without checking the retention criteria. You can add to or edit the DELETE statements to remove DBD and PSB instances that would not otherwise be eligible for deletion. You can code DELDBVER statements to remove all instances of a specified DBD version.

The update function of the utility sets or modifies the retention criteria in the header segment of individual DBD and PSB records. The retention criteria in a header segment overrides any default retention criteria that is specified the DFSDFxxx member of the IMS PROCLIB data set. If the update and analysis functions are requested at the same time, the utility performs the updates before performing the analysis.

The DFS3PU10 utility issues a CHKP (checkpoint) call to commit updates to the IMS catalog after every 200 updates, and message DFS4518I is issued in the utility output.

The modes and functions of the utility are specified by the following control statements:

- MODE ANALYSIS | PURGE | BOTH
- DELDBVER
- UPDATE DBD | PSB
- DELETE
- MANAGEDACBS

The DFS3PU10 utility can run in a DL/I or DBB region or a BMP region.

Subsections:

- “Restrictions” on page 390
- “Prerequisites” on page 390
- “Requirements” on page 390
- “Recommendations” on page 390
- “Input and output” on page 390
- “JCL specifications” on page 390
- “SYSIN control statements” on page 391

© Copyright IBM Corp. 1974, 2024

- "SYSUT1 control statements" on page 395
- "Other usage information" on page 396
- "Example JCL" on page 396
- "Return codes" on page 397

Restrictions

No restrictions are documented for this utility.

Prerequisites

No prerequisites are documented for this utility.

Requirements

The DFS3PU10 utility requires access to the IMS.PROCLIB data set that contains the DFSDFxxx member that enables the IMS catalog and defines the default retention criteria for the records in the IMS catalog.

If you run the DFS3PU10 utility in a DLI region and the IMS catalog is shared, you must specify IRLM support in the EXEC parameters of the utility JCL.

Recommendations

First, run the utility to obtain the list of DBD and PSB instances that can be deleted based on the retention criteria in effect for each DBD and PSB record. Then, examine the list to ensure that no DBD or PSB instances are included that are still needed by your IMS applications. Finally, run the utility to purge the unneeded DBD or PSB instances.

Input and output

The DFS3PU10 utility accepts the following input:

- The analysis and update functions read control statements from the SYSIN DD statement.
- The purge function reads control statements from the SYSUT1 data set.
- The analysis function reads the records in the IMS catalog.
- Both functions read from the CATALOG section of the DFSDF*xxx* member of the IMS.PROCLIB data set.

The DFS3PU10 utility generates the following output:

- The analysis function writes DELETE statements to the SYSUT1 data set.
- The analysis function writes a list of the DBD and PSB instances that are eligible for deletion to the SYSPRINT data set.
- The purge function writes a list of the DBD and PSB instances that were deleted to the SYSPRINT data set.
- The update function updates the header segments of DBD and PSB records in the IMS catalog.
- The purge function deletes segments or records from the IMS catalog.

JCL specifications

EXEC statement

The DFSDF= parameter specifies the three-character suffix of your DFSDF*xxx* member in the IMS.PROCLIB data set.

```
//BATCH EXEC PGM=DFSRRC00,
// PARM=(DLI,DFS3PU10,DFSCP001,,,,,,,,,,Y,N,,,,,,,,,,,'DFSDF=xxx')
```

To specify IRLM support, you can code the PARM parameter on the EXEC statement as shown in the following example:

```
PARM=(DLI,DFS3PU10,DFSCP001,,,,,,,,,,Y,Y,irlmid,,,,,,,,,,'DFSDF=xxx')
```

To run the utility in a BMP region:

```
PARM=(BMP,DFS3PU10,DFSCP001,,,,,,,,,,imsid,,,,,,)
```

DD statements

STEPLIB DD
Points to IMS.SDFSRESL, which contains the IMS nucleus and required IMS modules. If STEPLIB is unauthorized by having unauthorized libraries that are concatenated to IMS.SDFSRESL, you must include a DFSRESLB DD statement.

DFSRESLB DD
Points to an authorized library that contains the IMS SVC modules. For IMS batch, SDFSRESL and any data set that is concatenated to it on the DFSRESLB DD statement must be authorized through the Authorized Program Facility (APF). This DD statement provides an authorized library for the IMS SVC modules, which must be in an authorized library. The JOBLIB or STEPLIB statement does not need to be authorized for IMS batch.

PROCLIB DD
Defines the IMS.PROCLIB data set that contains the DFSDFxxx member that defines the default retention criteria for the records in the IMS catalog.

IMS DD
Defines the IMS.PSBLIB and IMS.DBDLIB data sets.

IEFRDER DD
Defines the primary IMS log data set.

DFSVSAMP
Defines the buffer pool parameters data set.

SYSPRINT
If the utility is run with MODE ANALYSIS specified, the SYSPRINT data set contains a list of the names and timestamps of the DBD and PSB and PSB instances that are eligible for deletion based on the retention criteria that is currently in effect. If the utility is run with MODE PURGE or MODE BOTH specified, the SYSPRINT data set contains a list of the DBD and PSB instances that were deleted. The DCB parameters for the SYSPRINT data set are RECFM=FBA and LRECL=121, but you do not have to specify these parameters.

SYSIN
A physical sequential data set that contains the utility control statements that are read by the analysis, purge, and update functions of the DFS3PU10 utility. The DCB parameters for the SYSIN data set are RECFM=FB and LRECL=80, but you do not have to specify these parameters.

SYSUT1
A physical sequential data set that contains the DELETE control statements, DELDBVER control statements, or both that are read by the purge function of the utility. The analysis function of the utility writes DELETE statements to the SYSUT1 data set.

Except when MODE BOTH is specified, you can edit the contents of the SYSUT1 data set to modify the generated DELETE statements or to add DELETE statements and DELDBVER statements.

The DCB parameters for the SYSUT1 data set are RECFM=FB and LRECL=80, but you do not have to specify these parameters.

SYSIN control statements

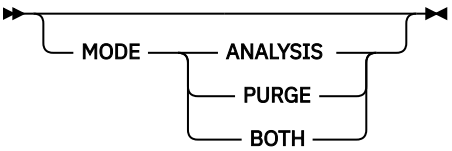
The utility accepts the following types of control statements through the SYSIN DD statement:

- MODE statement

- DELDBVER statement
- UPDATE statement
- MANAGEDACBS statement
- RESOURCE_CHKP_FREQ n

MODE statement

Specifies whether the utility executes the analysis function, the purge function, or both. The MODE statement can be specified only once.



ANALYSIS

Based on the retention criteria that is in effect for each record and any DELDBVER statements that are included as input, the utility generates DELETE statements for the DBD and PSB instances that are eligible for deletion. Nothing is deleted from the IMS catalog.

The retention criteria that are set in the HEADER segment of each catalog record is used to determine eligibility for deletion. If the HEADER segment of a record does not contain any retention criteria, the utility uses the retention criteria from the CATALOG section of the DFSDF*xxx* member of the IMS.PROCLIB data set.

The DELETE statements are written to the SYSUT1 data set, overwriting any existing contents. After the analysis is complete, you can review and, if necessary, edit the DELETE statements.

You can submit UPDATE statements with this mode. The UPDATE statements are processed before the utility examines the catalog HEADER information.

PURGE

The utility reads the DELETE statements in the SYSUT1 data set and purges the matching DBD and PSB instances from the database.

UPDATE statements are not allowed with this mode.

BOTH

The utility runs in ANALYSIS mode, determines which record instances are no longer needed according to the current retention criteria, and then runs in PURGE mode to remove the identified record instances.

You can submit UPDATE statements with this mode. The UPDATE statements are processed before the utility takes any other action.

DELDBVER statement

Deletes all instances of the specified version of a DBD. When DELDBVER is specified, MODE ANALYSIS or MODE BOTH must also be specified.

When MODE ANALYSIS is specified, the DELDBVER statement generates DELETE statements in the SYSUT1 data set for all of the DBD instances of the specified version number of a DBD, regardless of the retention criteria. The DBD instances that are proposed for deletion are also listed in the SYSPRINT data set.

When MODE BOTH is specified, the DELDBVER statement deletes all instances of the specified DBD version. The deleted DBD instances are listed in the SYSPRINT data set.

When MODE PURGE is specified, the DELDBVER statement cannot be specified in the SYSIN data set.

The DELDBVER statement can also be specified in the SYSUT1 data set. For more information, see “SYSUT1 control statements” on page 395.

The following diagram shows the syntax of the DELDBVER statement.

You can specify the following parameters on the DELDBVER statement:

member-name
The 8-character name of the DBD that you are deleting a version from.

version-number
The version number of the DBD that you are deleting. The value must match the version number that is specified on the DBVER keyword in the DBD generation statement of the version that you are deleting.

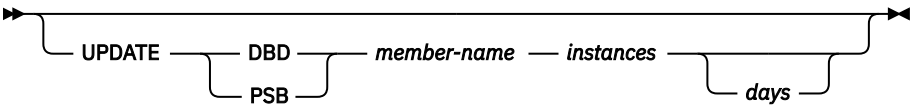
UPDATE statements

Use this statement to set the retention criteria for records in the catalog database. This information is stored in the HEADER segment of the relevant catalog record.

You can submit any number of UPDATE statements. If a MODE statement is specified with one or more UPDATE statements, the UPDATE statements are processed first.

If one or more UPDATE statements are specified without a MODE statement, the utility does not write DELETE statements to the SYSUT1 data set or delete any DBD or PSB instances.

When MODE PURGE is specified, the UPDATE statement cannot be specified in the SYSIN data set.



UPDATE DBD|PSB
Specify DBD or PSB. You can specify multiple UPDATE statements in the utility input.

member-name
The IMS resource name. These names are 8 characters, and wildcards are supported in the following formats:

- You can update the retention criteria for all DBD or PSB resources by specifying only the wildcard operator (*)
- You can update the retention criteria for DBD or PSB records that match a prefix value by specifying the prefix and then the wildcard operator (ABC*)

Updates are processed in alphanumeric order. Later updates override earlier ones. For example, if the following UPDATE statements are submitted:

```
UPDATE DBD DB* 1 365
UPDATE DBD * 1 20
UPDATE DBD DBOHIDK5 10 813
```

And your IMS catalog database contains records for the following resources:

```
DBD CUSTDB
DBD DB1XYZ
DBD DB2XYZ
DBD DB3XYZ
DBD DBOHIDK5
DBD DBVHDJ05
DBD EMHDB1
```

The utility processes the universal wildcard update first, assigning all DBD records the retention values of VERSIONS=1 and DAYS=20. Then it processes the statement for DB*, assigning all records with the DB prefix the retention values of VERSIONS=1 and DAYS=365. Finally, it processes the statement for DBOHIDK5, assigning that specific resource the retention values of VERSIONS=10 and DAYS=813. The later updates override the earlier updates.

instances
The number of instances of a DBD or PSB that must be retained in the DBD or PSB record.

For DBD instances, if database versioning is used, this value is the number of instances of the DBD that must be retained for each version of the DBD.

If this parameter is set to 1, any DBD or PSB instance other than the most recent instance is eligible for deletion.

If the number of instances in the record or the DBD version is less than this value, no instances are eligible for deletion.

If the number of instances in the record or the DBD version exceeds this value, the oldest instances are eligible for deletion, but only if they are older than the *days* value, if it is set.

This value is stored in the RETNINST field of the HEADER segment in DBD and PSB records.

days
The number of days that an instance of a DBD or PSB must be retained before it can be purged from the DBD or PSB record. Only DBD and PSB instances older than this number of days are eligible for deletion.

If this parameter is omitted or set to 0 for a DBD or PSB record, the age of the DBD or PSB instances is not used as a retention criterion. This value is stored in the RETNDAYS field of the HEADER record segment.

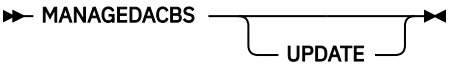
Note: When a value greater than 0 is specified on the *days* parameter, a DBD or PSB instance is eligible for deletion only if all of the following criteria are met:

- The age of the instance is greater than the *days* value
- The number of instances in the record or the DBD version is greater than the *instances* value
- The timestamp of the instance is older than the timestamps of the instances that are retained to satisfy the *instances* value

For example, if the number of DBD or PSB instances in a record is less than the *instances* value, no instances are eligible for deletion, even if the age of one or more of the instances is greater than the *days* value. Similarly, if the number of instances in a record is greater than the *instances* value, but the age of each instance in the record is less than the *days* value, no instances are eligible for deletion.

MANAGEDACBS
Specifies whether deleting DBD and PSB instances from the IMS catalog causes the corresponding DBD and PSB instances in the IMS directory to be deleted.

The MANAGEDACBS statement can be specified according to the following syntax diagram:



UPDATE
Deletes from the IMS directory the DBD and PSB instances that are deleted from the IMS catalog. The MANAGEDACBS UPDATE statement is automatically generated if the IMS directory is active, that is, if ACBMGMT=CATALOG is specified in the <CATALOG> section of the effective DFSDFxxx PROCLIB member. If both the MANAGEDACBS UPDATE and MODE ANALYSIS statements are specified, DBD and PSB instances are not deleted from the IMS directory.

RESOURCE_CHKP_FREQ n
Specifies the number of resource instances to be deleted or updated between checkpoints. *n* can be a 1- to 8-digit numeric value ranging from 1 to 99999999. The default value is 200.

SYSUT1 control statements

The control statements in the SYSUT1 data set are used by the DFS3PU10 utility to delete DBD and PSB segment instances, DBD versions, or whole DBD or PSB records from an IMS catalog.

The analysis function of the DFS3PU10 utility generates control statements in the SYSUT1 data set, which can then be used as input to the purge function.

If you specify MODE ANALYSIS, you can add to or edit the generated control statements in SYSUT1 before executing the purge function. However, if you specify MODE BOTH, you cannot review or edit the control statements before the purge function processes the SYSUT1 data set and deletes the segments and records.

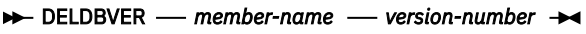
For processing when MODE PURGE is specified, you can code your own SYSUT1 data set or use the SYSUT1 data set that was produced by the analysis function in a previous execution of the utility.

The following control statements can be specified in the SYSUT1 data set.

DELDBVER statement

Deletes all DBD instances of the specified version of a DBD.

The following diagram shows the syntax of the DELDBVER statement.



You can specify the following parameters on the DELDBVER statement:

member-name
The eight-character name of the DBD from which you are deleting a version.

version-number
The version number of the DBD that you are deleting. The number that is specified here must match the version number that is specified on the DBVER keyword in the DBD generation statement of the version that you are deleting.

DELETE statements

Specifies a DBD or PSB instance or an entire DBD or PSB record to delete from the IMS catalog database.

Important: In a managed ACBs environment, active or pending directory members cannot be deleted when IMS is online if the utility is run as a BMP job.

The individual instances within each DBD or PSB record are differentiated by their DBD or PSB name and their ACB generation timestamps.

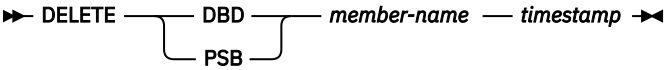
The analysis function of the utility automatically generates the DELETE statements based on the retention criteria that is currently in effect for each record and any DELDBVER statements in the SYSIN data set.

When MODE ANALYSIS is specified, you can review and edit the contents of the SYSUT1 data set before anything is deleted.

The eligibility of an instance for deletion is determined by the retention criteria that the utility reads from either the HEADER segment of a record or from the CATALOG section of the DFSDF*xxx* member of the IMS.PROCLIB data set.

During the purge mode, the utility does not check the retention criteria. If you manually code or edit the DELETE statements, you can remove DBD and PSB instances that would not otherwise be eligible for deletion.

The following syntax diagram shows the format of the DELETE statement:



DELETE DBD|PSB
You can delete either DBD or PSB resource record instances.

member-name
The IMS name of the DBD or PSB resource. These names are 8 characters, and wildcards are supported in the following formats:

- You can include all DBD or PSB resources by specifying only the wildcard operator (*).
- You can include DBD or PSB resources that match a prefix value by specifying the prefix and then the wildcard operator (ABC*).

timestamp
The ACB timestamp that identifies the specific DBD or PSB instance to purge.

The timestamp is in the following format: `yydddhhmmssth`.

Tip: You can specify a wildcard operator (*) to delete an entire DBD or PSB record, including the root segment and the header segments, from the IMS catalog.

Other usage information

If no retention criteria are stored in the RETNINST and RETNDAYS fields of the HEADER segment of a DBD or PSB record, the utility uses the DFSDF*xxx* member of the IMS.PROCLIB data set to determine the retention criteria.

If the RETNINST and RETNDAYS fields in the HEADER segment of the record both contain non-zero values, the utility does not use the values in the DFSDF*xxx* member to determine which DBD and PSB instances to purge. Instead, it uses the values from the RETNINST field (for the minimum number of instances) and the RETNDAYS field (for the minimum number of days) of the HEADER segment.

Attention: If the value of RETNINST is 0 and the value of RETNDAYS is non-zero, the utility generates a DELETE statement to purge all instances of the DBD or PSB from the record, including the instance that corresponds to the active member of the ACB library.

You might want to configure the DAYS parameter in the DFSDF*xxx* member to set a minimum number of days to retain catalog record instances, but selectively disable time-based retention for specific records. In that case, use this utility to explicitly specify a value of at least 1 for the number of instances and 0 for the number of days for those records.

For example, if the DFSDF*xxx* member contains the following retention information, no instances are eligible for deletion unless there are more than five instances in a record, and at least one of those instances is at least five days old.

```
RETENTION(INSTANCES=5,DAYS=5)
```

However, you can disable the time-based retention period for a subset of records in the IMS catalog by using the UPDATE statement of the DFS3PU10 utility to set the DAYS value to 0 directly in the HEADER segment of each record, as shown in the following example:

```
UPDATE DBD JK* 5 0
```

For each DBD record with the prefix JK, the preceding example sets RETNINST=5 and RETNDAYS=0. For these records, the IMS Catalog Record Purge utility does not consider age when determining eligibility for deletion. The utility generates DELETE statements for JK* DBD instances only if more than five instances are stored for in each JK* DBD record. The oldest record instances are removed first.

If no RETENTION values are specified in the CATALOG section of the DFSDF*xxx* member, the defaults are INSTANCES=2 and DAYS=0.

Example JCL

The following example of the utility JCL both updates the retention criteria and generates delete statements for all of the eligible DBD and PSB instances.

In the UPDATE phase of utility execution, the example JCL updates the retention settings for all PSB records and for the DBD records matching the prefix JK*.

In the ANALYSIS phase, the example JCL generates DELETE statements for all DBD and PSB instances that are eligible for deletion. The DELDBVER statement generates DELETE statements for all of the DBD instances of version 2 of the DBD JKDBA020. The utility writes the DELETE statements to the data set referenced by the SYSUT1 data set. You can then edit this data set or use it directly as SYSUT1 input to a separate run with MODE PURGE.

```
//BATCH EXEC PGM=DFSRRC00,
// PARM=(DLI,DFS3PU10,DFSCP001,,,,,,,,,,Y,N,,,,,,,,,,'DFSDF=001')
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//IMS      DD  DSN=IMS.PSBLIB,DISP=SHR
//         DD  DSN=IMS.DBDLIB,DISP=SHR
//SYSUT1   DD  ...             DELETE statements
//IEFRDER  DD  ...             Log data set
//DFSVSAMP DD  ...             Buffer pool parameters
//SYSPRINT DD  SYSOUT=*        Analysis or purge report
//SYSIN DD *                   Control statements
MODE ANALYSIS
UPDATE PSB * 5 365
UPDATE DBD JK* 5 365
DELDBVER JKDBA020 2
RESOURCE_CHKP_FREQ 300
```

The following JCL shows how you can code DELETE and DELDBVER statements of your own to use as SYSUT1 input to a run with MODE PURGE.

```
//BATCH EXEC PGM=DFSRRC00,
// PARM=(DLI,DFS3PU10,DFSCP001,,,,,,,,,,Y,N,,,,,,,,,,'DFSDF=001')
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//IMS      DD  DSN=IMS.PSBLIB,DISP=SHR
//         DD  DSN=IMS.DBDLIB,DISP=SHR
//IEFRDER  DD  ...             Log data set
//DFSVSAMP DD  ...             Buffer pool parameters
//SYSPRINT DD  SYSOUT=*        Analysis or purge report
//SYSIN DD *                   SYSIN control statements
 MODE PURGE
RESOURCE_CHKP_FREQ 300
/*
//SYSUT1   DD  *               SYSUT1 control statements
 DELETE DBD JKDBA600  1306514025679
 DELETE DBD  *       1300112161055
 DELETE PSB   *      1300112161055
 DELETE PSB  PSBJKE05    *
 DELDBVER JKDBA020 2
/*
```

Return codes

The IMS Catalog Record Purge utility completes execution with one of the following return codes:

0
No errors or exceptional conditions.

4
Warnings only.

8
Errors in control statements.

12
Invalid combinations of control statements, as indicated by messages DFS4424E or DFS4433E.

20
An error occurred that generated one of the following messages

- DFS4420E

- DFS4421E
- DFS4422E
- DFS4423E
- DFS4427E
- DFS4485E

24
The SYSPRINT DD statement was missing or invalid.

Related concepts
Removing DBD and PSB instances from the IMS catalog (Database Administration)
Related reference
CATALOG and CATALOGxxxx sections of the DFSDFxxx member (System Definition)

Part 3. Analysis utilities and reports

Use the analysis utilities to collect and format analyses and reports about the IMS system.

Each topic introduces how the utility works, defines requirements and restrictions for its use, and provides examples.

© Copyright IBM Corp. 1974, 2024

Chapter 15. Fast Path Log Analysis utility (DBFULTA0)

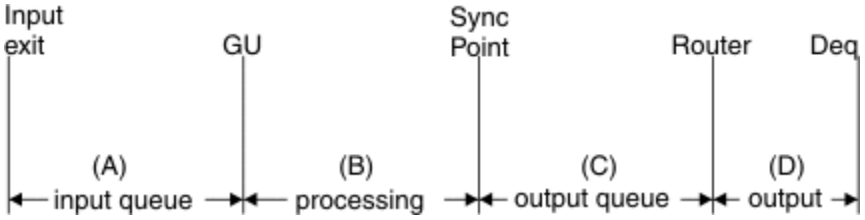
Use the Fast Path Log Analysis utility to prepare statistical reports for Fast Path, based on data that is recorded on the IMS system log.

This utility is an offline utility and produces three data sets, one of which contains seven formatted reports:

- Detail Listing of Exception Transactions
- Summary of Exception Detail by Transaction Code for IFP Regions
- Overall Summary of Transit Times by Transaction Code for IFP Regions
- Overall Summary of Resource Usage and Contentions for All Transaction Codes and PSBs
- Summary of Region Occupancy for IFP Regions by PST
- Summary of VSO Activity
- Recapitulation of the Analysis

These reports are useful for system installation, tuning, and troubleshooting. This utility is not related to the IMS Monitor or the Log Transaction Analysis utility.

The following figure shows four intervals that are computed for a Fast Path transaction:



(A)
Input queue time—period from input exit to the get unique (GU) call of the application program

(B)
Processing time—period from the get unique (GU) call of the application program to sync point

(C)
Output queue time—period from sync point to entry to the output router

(D)
Output time—period from output router entry to dequeue time

Figure 35. Intervals for a Fast Path transaction

The maximum interval that can be recorded on the logs is 65.535 seconds. However, if in computing the time span to be reported, the fields overflow, 9999 is displayed in the report, to indicate a computational overflow. The fields of IN-Q, PROC, and OUTQ can represent 9.999 seconds at maximum.

The four intervals are computed and inserted into reserved fields in Fast Path log records and are thus made part of the normal logging procedure. Intervals (A) and (B) appear in the input message (X'5901') and the output message (X'5903') log records respectively. Intervals (C) and (D) appear in the dequeue log record (X'5936'). Synchronization point takes place at the boundary between intervals (B) and (C).

The Fast-Path-Log-Analysis report includes additional performance-related data items from the Fast Path log records. The kinds of data items contained in the log records that might be reported are:

- Input message (X'5901') log record
  - The routing code for the transaction
  - The input terminal's LTERM name for the transaction
  - The balancing group queue count

© Copyright IBM Corp. 1974, 2024

- Synchronization point (X'5937') log record
  - The number of VSO reads
  - The number of VSO updates (CIs)
  - The number of ADS reads
  - The number of ADS updates (CIs)
  - The number of DEDB calls made
  - The number of MSDB calls made
  - The number of control interval (CI) contentions
  - The number of unit of work (UOW) contentions
  - The number of common buffers used
  - The number of waits for common buffers
  - The number of waits for private buffers

The intervals (A), (B), (C), (D), and the performance-related items are combined with other logged information to produce all the reports.

Subsections:

- "Restrictions" on page 402
- "Prerequisites" on page 402
- "Requirements" on page 402
- "Recommendations" on page 402
- "Input and output" on page 402
- "JCL specifications" on page 403
- "Utility control statements" on page 404
- "Return codes" on page 408

Restrictions

The Fast Path Log Analysis utility cannot use Common Queue Server (CQS) logs as input because CQS log records have a different format from IMS log records.

Prerequisites

Currently, no prerequisites are documented for the DBFULTA0 utility.

Requirements

Currently, no requirements are documented for the DBFULTA0 utility.

Recommendations

Currently, no recommendations are documented for the DBFULTA0 utility.

Input and output

The Fast Path Log Analysis utility uses the following input:

- An IMS system log data set

- A control statement that contains the execution parameters

The Fast Path Log Analysis utility processing consists of the following two steps:

1. Constructing Fast Path transaction detail records (FPTDR)
2. Analyzing the FPTDRs and printing the reports

The basic unit of output from this utility is the FPTDR. One FPTDR is constructed for each Fast Path transaction processed. An FPTDR is a 143-byte EBCDIC logical record consisting of the data associated with a given transaction (compiled from one or more log records) and a sequence number that indicates the order in which this transaction entered sync-point processing. The last log record that can supply data for each FPTDR is the dequeue record for the transaction.

The basic FPTDR record is extended to 252 bytes when written to the Exception Traffic data set. The first 143 bytes are identical to the Total Traffic data set.

The Fast Path Log Analysis utility uses the FPTDRs to form the following three output data sets:

- Total Traffic, normally a tape or direct-access data set that contains every FPTDR. This data set can be passed to a subsequent job step for sorting and printing, or for additional analysis.

  This data set is optional.
- Exception Traffic, normally a direct-access or tape data set that contains only those FPTDRs that you have set as exceptional and that therefore appear in the Detail Listing of Exception Transactions report. This data set can be passed to a subsequent job step for sorting and printing, or for additional analysis.

  This data set is optional.
- Formatted Reports, normally a printer output data set that consists of several reports formed by various combinations of transaction detail records.

The Total Traffic and Exception Traffic data sets are provided to make it convenient for you to post process performance data, formatted by the utility, without using the log data set. For example, inspection of reports can indicate that the Total Traffic data set should be sorted and printed in physical line number and terminal sequence, to analyze a problem possibly related to line activity. An internal DSECT within the source code for DBFULTA0, FPDR, maps these records.

Records are written to the Total Traffic and Exception Traffic data sets in the order in which they are completed—in the order of dequeue records for the normal transaction sequence. However, the sequence number assigned for each transaction is determined by the order in which the transaction enters sync point processing.

JCL specifications

The Fast Path Log Analysis utility executes as a standard operating system job. You must define an EXEC statement, DD statements and Utility control statements defining input and output.

EXEC statement

Executes the Fast Path Log Analysis utility.

```
//EXEC PGM=DBFULTA0
```

DD statements

STEPLIB DD
Describes the program library that contains the DBFULTA0 load module.

```
//STEPLIB DD DSN=IMS.SDFSRESL,DISP=SHR
```

SYSPRINT DD
Describes the data set that receives the printed output of DBFULTA0—reports, messages, and parameter statement images. This DD statement is required.

```
//SYSPRINT DD SYSOUT=A
```

SYSUT1 DD

Describes the data set that receives the total traffic output of DBFULTA0. This is a sequential data set consisting of every Fast Path transaction detail record formed by DBFULTA0. Each record is in EBCDIC characters. The logical record length is 143 bytes. The block size specification is optional. The default value for BLKSIZE is 1430.

```
//SYSUT1    DD DSN=&&TOTAL,DISP=(,PASS),UNIT=SYSDA,
//             SPACE=(CYL,(1,1)),DCB=BLKSIZE=2860
```

SYSUT2 DD

Describes the data set that receives the exception traffic output of DBFULTA0. This is a sequential data set consisting of the Fast Path transaction detail records that are exceptions. It is a copy of the Detail Listing of Exception Transactions with headings and carriage control characters suppressed. The logical record length is 252 bytes. The block size specification is optional. The default value of BLKSIZE is 2520.

```
//SYSUT2    DD DSN=&&EXCEP,DISP=(,PASS),UNIT=SYSDA,
//             SPACE=(CYL,(1,1)),DCB=BLKSIZE=5040
```

LOGTAPE DD

Describes the input log data set.

```
//LOGTAPE   DD DSN=IMS33.LOG,DISP=OLD,VOL=SER=XXXXXX,UNIT=XXXX
```

SYSIN DD

Describes the input control data set. This data set is used to specify execution parameters. This DD statement is optional. The following is a sample input stream.

```
//SYSIN DD *
START=09:59:59          24-hour notation, note colons
END=12:00:00
LINECNT=45              lines per page for reports
NOT-MESSAGE             include transactions that are not IFPs
MAXDETAIL=5000          exceptions detail listing limit
CALLS
BUFFER
VSO
TT(*)=15.0
TT(TCODE1)=3.0
TT(TCODE2)=2.5
TT(TCODE3)=1.0
```

Utility control statements

Control statements in the SYSIN data set control the Fast Path Log Analysis utility. You can specify the time period of Fast Path execution for which the analysis is to be performed. This is expressed as the starting time (clock time) or an ending time. Transactions whose synchronization point time stamps fall within this interval are processed. If you do not specify an interval, the entire log data set is processed.

After the log is processed up to the end time specified, scanning continues to find dequeue records related to transactions that were processed during the specified analysis time interval.

Process multi-volume log data sets by specifying multiple volumes in the //LOGTAPE DD statement or by concatenation of DD statements.

Transit time exception specification

You can limit the volume of printed output produced by specifying an exceptional transit time value for each transaction code. Occurrences of transaction transit times that are less than the exceptional value do not appear in the Detail Listing of Exception Transactions. You can specify a different exception transit time for each unique transaction code. Also, you can specify a global value for all transaction codes that

are not individually specified. A separate summary report is produced for those transactions that exceed the exception criteria.

A detail report of all the transactions processed from the log data set can be produced either by not specifying an exceptional transit time (default=0) or by printing the total FPTDR data set in a subsequent job step.

An upper limit can be placed on the number of transactions that are printed in the Detail-Listing-of-Exception-Transactions report. This limit can be used to prevent the production of unexpectedly large output listings.

Analysis parameter statement formats

All statements begin in column 1. The statements can appear in any order and are listed in the SYSPRINT data set for verification.

Starting date specification (STARTDAY)

You can specify the date of the earliest transaction to be processed in Julian format. Transactions with an earlier date are ignored. If the starting time is also specified, transactions with an earlier synchronization point time on that day are also ignored. The format of this parameter is:

```
STARTDAY=YYDDD
```

YYDDD is the last two digits of the year and the sequential number of the day, running from 1 to 366.

The default value is the date IMS was started, from the type X'42' log record.

The IMS Fast Path Log Analysis utility (DBFULTA0) expects the STARTDAY= parameter if the START= parameter is provided.

Ending date specification (ENDDAY)

You can specify the date of the latest transaction to be processed in Julian format. Transactions with a later date are ignored. If the ending time is also specified, transactions with a later synchronization point time on that day are also ignored. The format of this parameter is:

```
ENDDAY=yyddd
```

(*yy* represents the last two digits of the year, and *ddd* represents the sequential number of the day, from 001 to 365)

The default value, if ending time is specified, is the date IMS was started from the type X'42' log record. If ending time is less than starting time, the default is one day later. If neither ending date nor ending time are specified, the entire data set is processed.

The DBFULTA0 utility expects the ENDDAY= parameter if the END= parameter is specified.

Starting time specification (START)

You can specify the time of the earliest transaction to be processed. Transactions with an earlier sync-point time are ignored. The format of this parameter is (in hours, minutes, and seconds for a 24-hour clock):

```
START=HH:MM:SS[{+|-}HH:MM]
```

You only need to specify the optional time-zone information if the offset to the Universal Time Coordinated on the day entered is different from the current offset, for example because of a daylight savings time change.

The optional time-zone information following hh:mm:ss contains the following:

+ or -
Specifies the sign of the time-zone offset from UTC.

HH
Specifies the number of whole hours of offset from UTC.

MM
Specifies minutes of offset. MM can be 00, 15, 30, 45, or blank.

The default value is 00:00:00, which causes the analysis to begin with the first transaction on the log data set.

Ending time specification (END)

You can specify the sync-point time of the latest transaction to be processed. Transactions with a later synchronization point time will be ignored. The format of this parameter is (in hours, minutes, and seconds for a 24-hour clock):

```
END=HH:MM:SS[{+|-}HH:MM]
```

You only need to specify the optional time-zone information if the offset to UTC on the day entered is different from the current offset, for example because of a daylight savings time change.

The optional time-zone information following hh:mm:ss contains the following:

+ or -
Specifies the sign of the time-zone offset from UTC.

HH
Specifies the number of whole hours of offset from UTC.

MM
Specifies minutes of offset. MM can be 00, 15, 30, 45, or blank.

If the end date is not specified, the default value causes the analysis to end with the last transaction on the log data set.

The date on the log data set is not explicitly specified by a parameter statement. The data is implicit with the specification for the log data set that is in the JCL Requirements. The Julian date is read from the log header record when execution begins, and this date is printed as part of the parameter summary for verification.

Exceptional transit time specification (TT)

You can specify a time interval for each Fast Path transaction that you decide to consider exceptional for reporting purposes. The format of this parameter is (in seconds and tenths of seconds):

```
TT (TRANCODE)=SS.T
```

The transaction code, up to eight characters, is enclosed in parentheses. You can specify as many as 100 individual transaction codes. A global value of exceptional transit time is specified as follows: TT(*)=SS.T (in seconds and tenths of seconds).

This value applies to all transaction codes that are not individually specified. Individual specification overrides the global value. The default value for the global exceptional transit time is 0. A practical upper limit of exceptional transit time is 65.5 seconds. This limitation results from the field size used to express the time intervals (A), (B), and (C) in the Fast Path log records.

Not message-driven option (NON-MESSAGE or NOT-MESSAGE)

You can specify that transactions that are not IFPs (that is, BMPs, MPPs, utilities and DBCTL threads) should be considered exceptions and be included in the Detail-Listing-of-Exception-Transactions report. The accepted formats are:

```
NON-MESSAGE
```

or

```
NOT-MESSAGE
```

Both formats have the same result.

Detail-Listing-of-Exception-Transactions report size limitation (MAXDETAIL)

You can limit the number of lines printed in the Detail Listing of Exception Transactions. After this limit is reached, the analysis continues; however, no further transactions are printed in the Detail Listing of Exception Transactions.

The format of this parameter is:

```
MAXDETAIL=n
```

where n is an integer of no more than seven digits. The default value is 1000. The limitation of printed output lines does not affect the number of exception detail records that are written to the exception detail traffic data set (SYSUT2).

DL/I call specification (CALLS)

You can specify that the number of DL/I calls be printed. They are printed by call type (GU, REPL, and so on). The format of this parameter is:

```
CALLS
```

Information about calls is obtained from type X'5937' log records.

Buffer use specification (BUFFER)

You can specify that the amount of buffer use, by type, be printed. The format of this parameter is:

```
BUFFER
```

The following information is collected about buffer use:

- The number of NBA buffers used (NBA)
- The number of overflow buffers used (OVFN)
- The number of times buffer stealing was invoked by this transaction (STEAL)
- The number of times the transaction waited for a buffer to become available (WAIT)
- The number of buffers sent to OTHREAD (OTHR)
- The number of buffers used by MSDB and SDEP processing (NRDB)

Information about buffer use is obtained from type X'5937' log records.

64-bit Fast Path buffer pool specification (FPBP64)

You can specify that the amount of 64-bit Fast Path buffer pool use, by type, be printed. The format of this parameter is:

```
FPBP64
```

The following information is collected about 64-bit Fast Path buffer pool use for pool sizes 512 through 28 KB:

- The number of common buffers obtained by the thread during DL/I call processing
- The number of system buffers obtained by the thread during DL/I call processing
- The number of common buffers containing data which was written to a Fast Path database
- The number of system buffers containing data which was written to a Fast Path database
- The number of times a region waits for a buffer

Information about FPBP64 is obtained from type X'5945' log records.

Data space use specification (VSO)

You can specify that information on data space use, by transaction, be printed. The format of this parameter is:

```
VSO
```

The following information is collected about data space use:

- The number of CI read requests satisfied from a data space (VGET)
- The number of CIs with updates to a data space (VPUT) This number represents the number of CIs that would have been sent to OTHREAD if the areas were non-VSO.
- The number of CIs read from DASD into a data space (DGET)

Information about data space use is obtained from type X'5937' log records.

Printed page line count specification (LINECNT)

You can specify the number of lines printed per page for the printed reports. The format of this parameter is:

```
LINECNT=n
```

where n is an integer greater than 5. The value specified applies to titles and headers so that 6 is the minimum allowable value. The default value is 55 lines per page.

Each parameter statement is listed in the SYSPRINT data set exactly as it is read for verification. The following example shows parameter statements read from the SYSIN data set and of how they are listed in the SYSPRINT data set.

```
SPECIFIED INPUT PARAMETERS:

          ANALYSIS START TIME:  00:00:00     DATE: 2010187
                     END TIME:  23:59:59
          A MAXIMUM OF    1000 EXCEPTIONAL TRANSACTIONS WILL BE LISTED.
          RATE CALCULATION ACTIVE: INTERVAL=86399 SECONDS.
          TRANSIT TIME EXCEPTION VALUES:
                                          EXCEPTION VALUE IN SEC.
                    TRANSACTION CODE        (IN-Q THRU OUT-Q)

                    ----------_----         ------------------
                        *GLOBAL*                    0.0
```

After all parameter statements are read, the utility prints a summary display of either the parameters supplied or the default values that are used for parameters not specified. If you specify both the START and END parameters, then the line RATE CALCULATION ACTIVE will be displayed, and the Summary of Region Occupancy Report will be generated. The following example shows the parameter display. Date information is obtained from the log buffer control record (X'42').

```
LOG DATA SET ANALYSIS FOR IMS FAST PATH
PAGE   1
THE FOLLOWING PARAMETER CARDS WERE READ FROM SYSIN:
LINECNT=45
```

Return codes

User abend codes are not generated.

The following return codes are produced:

Code
Meaning

0
Successful completion of analysis

4
Analysis prematurely ended, partial results produced

8
Unable to perform analysis

12
Unable to open ddname SYSPRINT

Related reference
Log records (Diagnosis)

Fast Path report types

Fast Path reports provide details about transactions.

Format of total traffic and exception traffic data sets

The Fast Path Log Analysis utility gathers the Fast Path transaction detail records that are written to the Total Traffic and Exception Traffic data sets. A single logical record is written for each FPTDR. The data set organization is fixed blocked, with LRECL=143 for SYSUT1 (the Total Traffic data set) and LRECL=252 for SYSUT2 (the Exception Traffic data set). The BLKSIZE can be specified in the //SYSUT1 and //SYSUT2 DD statements, however, the default blocking factor is 10. The logical records are written in order of the dequeuing record associated with each transaction. The data code is EBCDIC for all characters. The format of each logical record is mapped by internal DSECT FPDR.

All leading zeroes of the edited fields are suppressed; however, there is always at least a single nonblank digit to the left of a decimal point.

Fields that are unused (for example, the output time field of a record that has no dequeue information) are set to blanks.

The synchronization point date and IMS release level fields are included in the SYSUT1 and SYSUT2 data sets for informational purposes, but will not appear on formatted reports.

Decimal integer fields that contain overflow values are indicated by the value of all 9s. This method of indicating overflow causes overflowed fields to sort high.

Detail-Listing-of-Exception-Transactions report

You define, with input parameters, what is considered to be an exceptional transit time value (TT input parameter) for each IFP transaction. Transit time is defined as the sum of intervals A, B, and C (defined in Figure 35 on page 401). Output time D is not included for this purpose. Any transaction with a transit time that exceeds the specified exceptional value is included in the Detail Listing of Exception Transactions and can be written on the SYSUT2 data set.

The following transactions are included in the report:

- Successfully processed IFP transactions with a transit time equal to or greater than the Exceptional Transit Time Specification.

  These include transactions for which a dequeue log record is not found. For these transactions the output queue time, and therefore the total transit time, are unknown and are not formatted. This condition is marked in the report by the characters NO DEQ under the TOTAL column.
- All IFP transactions with a synchronization point failure. These include invalid work prior to the first message GU and invalid work done after a message GU has received a ‘QC' status code, or if the transaction returns to IMS without receiving a ‘QC' status code.
- If you specify the "nonmessage" option, non-message-driven transactions are included.

You can limit the actual number of transactions reported with the MAXDETAIL input parameter. CALLS, BUFFER, and VSO lines are omitted for transactions that are not processed at the IMS for which the Fast Path Log Tape Analysis utility is run.

The following figure is an example of a Detail-Listing-of-Exception-Transactions report.

```
DETAIL LISTING OF EXCEPTION TRANSACTIONS:                                                
```

SYNC POINT TIME
The clock time at synchronization point processing.

S F
Synchronization failure reason code character for transactions that fail synchronization processing. A nonblank character in this column indicates synchronization failure and, in the preceding figure, the columns are blank. The following figure shows an example of a report with transactions that failed synchronization processing.

```
DETAIL LISTING OF EXCEPTION
TRANSACTIONS:                                                                                    PAGE   4


   SEQ  TRANCODE SYNC POINT  S ROUTING  LOGICAL  PST QUEUE TRANSIT TIMES(MSEC)- -OUT- DEDB ..ADS.. ..VSO.. MSDB BUF
CONTENTIONS R P
   NO.   OR PSB     TIME     F  CODE    TERMINAL  ID COUNT IN-Q PROC OUTQ TOTAL (SEC) CALL  RD UPD  RD UPD CALL USE CI
UW OB BW T T
------- -------- ----------- - -------- -------- --- ----- ---- ---- ---- ----- ----- ---- --- --- --- --- ---- --- --
-- -- -- - -
     1 PBVDSAGR 16:26:20.55                      1                                    32  16   0  16  16    0  16  0
0  0  0 B
       CALLS  - GU   0  GN   0  GNP   0  GHU   0  GHN   0  GHNP   0  REPL   0  ISRT  32  DLET   0  FLD   0  POS   0
TOTAL   32
       BUFFER - NBA=  16 OVFN=   0 STEAL=   0 WAIT=  0  OTHR=   0 NRDB=   0 PBUF=   0 PBWT=   0 ASIO=   0 AIOW=
0
       VSO    - VGET    0 VPUT   16 DGET
16
     6 BMP255   16:27:08.37 L                    1                                    42  13   0   0   0    0  10  0
0  0  0
       CALLS  - GU   0  GN   0  GNP   0  GHU   0  GHN  21  GHNP   0  REPL   0  ISRT   0  DLET  21  FLD   0  POS   0
TOTAL   42
       BUFFER - NBA=   5 OVFN=   5 STEAL=   4 WAIT=  0  OTHR=   0 NRDB=   0 PBUF=   0 PBWT=   0 ASIO=   0 AIOW=
0
       VSO    - VGET    0 VPUT    0 DGET
0
     7 BMP255   16:27:08.38 R                    1                                    42  13   0   0   0    0  10  0
0  0  0
       CALLS  - GU   0  GN   0  GNP   0  GHU   0  GHN  21  GHNP   0  REPL   0  ISRT   0  DLET  21  FLD   0  POS   0
TOTAL   42
       BUFFER - NBA=   5 OVFN=   5 STEAL=   4 WAIT=  0  OTHR=   0 NRDB=   0 PBUF=   0 PBWT=   0 ASIO=   0 AIOW=
0
       VSO    - VGET    0 VPUT    0 DGET
0
```

The meaning of nonblank codes A through U is as follows:

A
MSDB verify failure

B
MSDB arithmetic overflow

C
DEDB sequential dependent area full

D
DEDB sequential dependent insert caused buffer overflow

E
DEDB sequential dependent buffer overflow three times

F
DEDB area not available for use

G
Dynamic MSDB area full

H
MSDB required segment not found

I
DEDB FLD calls; lock for a CI could not be obtained

J
DEDB FLD calls; deadlock occurred

K
DEDB FLD calls; overflow occurred

L
ROLB call

M
DEDB FLD calls; verify failed

N
DEDB FLD calls; segment in CI was deleted

O
Out of resources

P
Inflight condition in /ERE

Q
RESYNC abort requested

R
Resource deadlock

S
Out of space in data sets

U
Application program abend

Information relating to sync failures is obtained from type X'5938' log records.

ROUTING CODE
Identification of the balancing group.

LOGICAL TERMINAL
The input LTERM name for this transaction.

PST-ID
The PST number.

QUEUE COUNT
The number of transactions in the balancing group (BALG) queue when this transaction entered synchronization point processing.

Transit Times in Milliseconds

IN-Q
Time interval A, input queue time in milliseconds.

The input queue time will be marked N/A for Shared EMH input/output transit time when the transaction is:

1. Local only
2. Global only or local first transaction which is processed on other CPC while DBFULTA0 is reading the log of the IMS backend.

PROC
Time interval B, processing time in milliseconds.

OUTQ
Time interval C, output queue time in milliseconds. Information relating to output queue time is obtained from type X'5936' log records, the terminal output dequeue records.

The input queue time will be marked N/A for Shared EMH input/output transit time when the transaction is:

1. Local only
2. Global only or local first transaction which is processed on other CPC while DBFULTA0 is reading the log of the IMS backend.

TOTAL
The sum of time intervals A, B, C. This is the transit time as defined for the utility. The magnitude of this sum exceeds the exception value for the transaction code.

OUT TIME
Time interval D, output time (to dequeue) in seconds.

DEDB CALL
The total number of DEDB calls.

ADS READS & UPDATES
The number of CIs read and updated.

VSO READS & UPDATES
The number of CIs read and updated from the data space.

MSDB CALL
The number of MSDB calls during this processing.

BUF USE
The total number of buffers used from the common buffer pool. This number includes non-related buffers used for MSDBs and SDEPs.

CONTENTIONS

CI
The number of waits for CIs during this processing.

UW
The number of waits for UOWs during this processing.

OB
The number of waits for overflow buffer allocation. This number should never be greater than 1.

BW
The number of waits for common buffers.

RT
The region type, one of the following:

B
BMP

I
IFP

M
MPP

U
Utility

PT
The process type, one of the following:

G
Shared EMH global message processing

H
HSSP

R
Reorganization

The following lines are only obtained if the optional utility control statements are provided. However, the information is always available in the extension to the FPTDR record in the SYSUT2 data set.

CALLS Line
The CALLS line contains the number of DL/I calls by type for DEDB calls. Information relating to CALLS is obtained from type X'5937' log records.

The different types of DL/I calls are:

GU CALL
The number of GU calls

GN CALL
The number of GN calls

GNP CALL
The number of GNP calls

GHU CALL
The number of GHU calls

GHN CALL
The number of GHN calls

GHNP CALL
The number of GHNP calls

REPL CALL
The number of REPL calls

ISRT CALL
The number of ISRT calls

DLET CALL
The number of DLET calls

FLD CALL
The number of FLD calls

POS CALL
The number of POS calls

TOTAL
The number of DL/I calls during this processing

BUFFER Line
The BUFFER line contains the amount of buffer use by type. Information relating to BUFFER is obtained from type X'5937' log records:

The different types of buffer use are:

NBA
The number of times a wait for NBA latch occurred during this processing.

OVFN
The number of overflow buffers used during this processing.

STEAL
The number of times buffer stealing is invoked by this transaction.

WAIT
The number of times the transaction waited for a buffer to become available.

OTHR
The number of buffers sent to OTHREAD.

NRDB
The number of buffers used by MSDB and SDEP processing.

PBUF
The number of private buffers used by HSSP or the High Speed DEDB Direct Reorganization utility in a transaction (one unit of work).

PBWT
The number of waits for private buffers by HSSP or the High Speed DEDB Direct Reorganization utility in a transaction (one unit of work).

ASIO
The number of UOW asynchronous read-aheads by HSSP or the High Speed DEDB Direct Reorganization utility in a transaction (one unit of work).

AIOW
The number of UOW asynchronous read-aheads to complete by HSSP or the High Speed DEDB Direct Reorganization utility in a transaction (one unit of work).

This number should be either zero or one.

VSO Line
The VSO line contains information on data space use by transaction. Information relating to VSO is obtained from type X'5937' log records.

The type of information collected about data space use is as follows:

VGET
The number of CI read requests satisfied from a data space.

VPUT
The number of CIs with updates to a data space.

This number represents the number of CIs that would have been sent to OTHREAD if the areas were non-VSO.

DGET
The number of CIs read from DASD into a data space.

SEMHB Line
The SEMHB line contains the transit time for Fast Path input and output messages on EMHQ. Information relating to SEMHB is obtained from type X'5936' log records.

The type of information collected about data space use is as follows:

SHARED EMHB
Shared EMH global message processing.

IMSG TRANSIT
The time that a Fast Path input message spent on the EMHQ before an application GU. The time is in milliseconds.

OMSG TRANSIT
The time that a Fast Path output message spent on the EMHQ before an application GU. The time is in milliseconds.

You can specify exceptional transit time values separately for each Fast Path transaction code. A global value can be specified that applies to all other unspecified transaction codes.

Summary-of-Exception-Detail-by-Transaction-Code (for IFP Regions) report

A summary is produced for the exceptional transactions selected for the Detail Listing of Exception Transactions. However, only the exceptional IFP transactions are taken into account. None of the other transaction types are included even if the NON-MESSAGE option is specified.

Transactions for which a dequeue record was not found are not included in this summary.

The following figure is an example of the Summary-of-Exception-Detail-by-Transaction-Code report.

```
 SUMMARY OF EXCEPTION DETAIL BY TRANSACTION CODE FOR IFP
REGIONS                                          PAGE   6
                  -------------- TRANSIT TIMES IN MILLI-SECONDS --------------    INPUT  MSG
OUTPUT MSG
 TRANS      -NO.OF-   ----TOTAL---    --INPUT Q --    --PROCESS --    --OUTPUT Q--    LENG  (CH)    LENG
(CH)-
 CODE       -TRANS-   -AVG-  -MAX-    -AVG-  -MAX-    -AVG-  -MAX-    -AVG-  -MAX-    -AVG  -MAX
-AVG  -MAX
 --------   -------   -----  -----    -----  -----    -----  -----    -----  -----    ----  ----
----  ----
 TPCA       157837     381    889      293    682       40    405       47    325      94    94
100   100
```

The column headings for this report are:

TRANS CODE
The transaction code.

NO. OF TRANS
The number of occurrences of the transaction code for which a transit time value was computed.

TRANSIT TIMES
The average and maximum values of transit time intervals in milliseconds.

INPUT LENG
The average and maximum values of input message length.

OUTPUT LENG
The average and maximum values of output message length.

The averages are computed using the number of occurrences of the transaction code for which a transit time value was computed.

Overall-Summary-of-Transit-Times-by-Transaction-Code (for IFP-Regions) report

A summary report is produced, by transaction code, for all IFP transactions found for the analysis period. Transactions for which a dequeue record was not found are not included in the summary.

The format of this report is identical to that of the Summary of Exception Detail by Transaction Code for IFP Regions. The following figure is an example of the overall summary of transit times by transaction code for IFP regions.

```
 OVERALL SUMMARY OF TRANSIT TIMES BY TRANSACTION CODE FOR IFP
REGIONS:                                          PAGE   7
                     -------------- TRANSIT TIMES IN MILLI-SECONDS --------------    INPUT  MSG
OUTPUT MSG
 TRANS      -NO.OF-   ----TOTAL---    --INPUT Q --    --PROCESS --    --OUTPUT Q--    LENG  (CH)    LENG
(CH)-
 CODE       -TRANS-   -AVG-  -MAX-    -AVG-  -MAX-    -AVG-  -MAX-    -AVG-  -MAX-    -AVG  -MAX
-AVG  -MAX
 --------   -------   -----  -----    -----  -----    -----  -----    -----  -----    ----  ----
____ ____
 TPCA        157837     381    889      293    682       40    405       47    325      94    94
100   100
```

Overall Summary of Resource Usage and Contentions for All Transaction Codes and PSBs report

A summary report is produced for all transactions and PSBs that had their synchronization point processing during the interval specified for the analysis. These include successfully processed and failed transactions from MPP, BMP and utility regions, and DBCTL threads. Data is summarized by PSB name or transaction code.

The following figure is an example of the overall summary of resource usage and contentions for all transaction codes and PSBs.

```
 OVERALL SUMMARY OF RESOURCE USAGE AND CONTENTIONS FOR ALL TRANSACTION CODES AND PSBS:                        PAGE   8
 TRANCODE --NO.-- ------DEDB CALLS------- -MSDB-- ----ADS I/O---- ----VSO ACT---- -COMMON BUFFER- TOTL CONTENTIONS TRAN
LGNR STATS
 --OR---- ---OF-- -TOTAL- --GET-- --UPD-- -CALLS- --RDS-- --UPD-- --RDS - --UPD-- -----USAGE----- SYNC TOT TOT CI/ RATE
-NO. OF CI
 --PSB--- -TRANS- AVG MAX AVG MAX AVG MAX AVG MAX AVG MAX AVG MAX AVG MAX AVG MAX AVG MAX WTS STL FAIL UOW OBA SEC /SEC
COMB LOG'D
 -------- ------- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ----- --- --- --- ----
_____ _____
 TPCA     157837   5   5   1   1   2   2   0   0   3   3   1   1   4   4   2   2   5   5   0   0    0   0   0 106
1315    0     0
```

The column headings of this report are:

TRANCODE OR PSB
The transaction code or PSB.

NO. OF TRANS
The number of occurrences of the transaction code for which a transit time value was computed.

DEDB CALLS
The number of DEDB calls

TOTAL
The total number of DL/I calls during this processing

GET
The total number of "GET" DL/I calls during this processing (GU, GN, GNP, GHU, GHN, GHNP)

UPD
The total number of "UPDATE" DL/I calls during this processing (REPL, ISRT, DLET, FLD)

AVG
The average number of calls per processing interval

MAX
The maximum number of calls per processing interval

MSDB CALLS (AVG MAX)
The average and maximum numbers of MSDB calls per processing interval.

ADS I/O
The area data set I/O

RDS
The total number of "READ" DL/I calls (GU, GN, GNP, GHU, GHN, GHNP) during this processing for an area data set

UPD
The total number of "UPDATE" DL/I calls (REPL, ISRT, DLET, FLD) during this processing for an area data set

AVG
The average number of calls per processing interval

MAX
The maximum number of calls per processing interval

VSO ACT
The amount of VSO activity

RDS
The total number of CI read requests satisfied from a data space

UPD
The total number of CIs with updates to a data space

AVG
The average number of calls per processing interval

MAX
The maximum number of calls per processing interval

COMMON BUFFER USAGE
The amount of buffer usage

AVG
The average number of calls per processing interval

MAX
The maximum number of calls per processing interval

WTS
The total number of times a transaction waited for a buffer to become available

STL
The total number of times buffer stealing was invoked for the transaction

TOTL SYNC FAIL
The total number of occurrences of this transaction code that failed synchronization point processing.

CONTENTIONS
The number of control interval contentions

TOT UOW
The total number of times unit-of-work contentions occurred for this transaction code

TOT OBA
The total number of times overflow buffer area contentions occurred for this transaction code

CI/SEC
The total number of CI contentions per second for this transaction code. If the time interval is less than one second, then it will default to one second

TRAN RATE/SEC
The average transaction rate for this transaction code. If the time interval is less than one second, then it will default to one second

LGNR STATS
The statistics related to the LGNR specification

NO. OF CI COMB
The total number of times the LGNR specification was exceeded for this transaction code. This number is either 0 or 1.

NO. OF CI LOG'D
The total number of times an entire CI was logged for this transaction code. This number is either 0 or 1 and will only be 1 if "NO. OF CI COMB" is also 1.

Summary-of-Region-Occupancy report

A summary report is produced of approximate region occupancy for IFP regions during a specified period. If the time interval is less than one second, then it defaults to one second. This information can be used to determine if an appropriate number of IFP regions are available for processing the workload.

This report is generated only if both the START and END parameters are specified for the utility. The following example shows the sample summary of region occupancy (percent) for IFP regions by PST.

```
SUMMARY OF REGION OCCUPANCY (PERCENT) FOR IFP REGIONS BY
PST                                      PAGE   9

     MEASUREMENT INTERVAL=    120 SECONDS.

REGION   1 HAD 70% OCCUPANCY WITH  84.4 SEC OF TOTAL PROCESS TIME DURING    978
TRANSACTIONS. RELATED PSB=TPC
REGION   2 HAD 67% OCCUPANCY WITH  81.1 SEC OF TOTAL PROCESS TIME DURING    922
TRANSACTIONS. RELATED PSB=TPC
REGION   3 HAD 68% OCCUPANCY WITH  82.2 SEC OF TOTAL PROCESS TIME DURING    956
TRANSACTIONS. RELATED PSB=TPC
REGION   4 HAD 67% OCCUPANCY WITH  81.3 SEC OF TOTAL PROCESS TIME DURING    926
TRANSACTIONS. RELATED PSB=TPC
REGION   5 HAD 69% OCCUPANCY WITH  83.4 SEC OF TOTAL PROCESS TIME DURING    972
TRANSACTIONS. RELATED PSB=TPC
REGION   6 HAD 67% OCCUPANCY WITH  80.7 SEC OF TOTAL PROCESS TIME DURING    919
TRANSACTIONS. RELATED PSB=TPC
REGION   7 HAD 70% OCCUPANCY WITH  84.1 SEC OF TOTAL PROCESS TIME DURING    978
TRANSACTIONS. RELATED PSB=TPC
REGION   8 HAD 68% OCCUPANCY WITH  82.5 SEC OF TOTAL PROCESS TIME DURING    942
TRANSACTIONS. RELATED PSB=TPC
REGION   9 HAD 66% OCCUPANCY WITH  80.4 SEC OF TOTAL PROCESS TIME DURING    944
TRANSACTIONS. RELATED PSB=TPC
REGION  10 HAD 70% OCCUPANCY WITH  84.8 SEC OF TOTAL PROCESS TIME DURING    958
TRANSACTIONS. RELATED PSB=TPC
```

Note: The PSB name may be "********" on the report if there is insufficient data on the log to determine the PSB name.

Summary-of-VSO-Activity report

A summary report is produced of VSO performance statistics by area. This report is generated only if there have been writes to the disk. The following figure is an example of this report.

```
SUMMARY OF VSO ACTIVITY                                PAGE  12

 SHR(0/1)   VSO       VSO       DASD      DASD      I/O
  AREA      GETS      PUTS      GETS      PUTS      SCHED
--------  --------  --------  --------  --------  --------

BRANCH01      8092      8095         0      6012      2154
TELLER01      8200      8198         0      8018      3752

 SHR(2/3)    CF       CF     READ  READ   DASD     DASD
  AREA      GETS     PUTS    HIT    XI    GETS     PUTS
--------  -------  -------  ____  ____  -------  -------
AREAFR01  1234567  1234567   99%   99%  1234567  1234567
AREA2     1234567  1234567   N/A   N/A  1234567  1234567
```

The column headings of the Summary-of-VSO-Activity report are:

VSO GETS
The total number of CI read requests satisfied from a data space.

VSO PUTS
The total number of CIs with updates to a data space. This number is the total number of CIs that would have been sent to OTHREAD if the areas were non-VSO.

DASD GETS
The number of CIs read from DASD into a data space.

DASD PUTS
The number of CIs written from a data space to DASD.

I/O SCHED
The total number of I/Os scheduled.

CF GETS
The total number of CI read requests satisfied by a coupling facility.

CF PUTS
The total number of CIs with updates to a coupling facility.

READ-HIT
The percentage of searches of the pool and the number of times that buffers were found. This is only valid for a lookaside pool.

READ-XI
The percentage of times a buffer was found in the pool and the number of times the buffer was invalid. This is only valid for a lookaside pool.

DASD GETS
The number of CIs read from DASD into the coupling facility.

DASD PUTS
The number of CIs written from the coupling facility to DASD.

Recapitulation-of-the-Analysis report

The following figure is an example of the recapitulation of the analysis report.

```
RECAPITULATION OF THE ANALYSIS:                                        PAGE  13
```

```
(1) TOTAL NUMBER OF FAST PATH TRANSACTIONS EXAMINED (SYSUT1).................157837
(2) NO. OF TRANSACTIONS INCLUDED IN THE EXCEPTION DETAIL DATA SET (SYSUT2)...157837
              BREAKDOWN BY EXCEPTION TYPE:
                   (2.1) TRANSIT TIME...................157837
                   (2.2) IFP SYNC FAILURE.....................0
                   (2.3) NO DEQUEUE RECORD....................0
                   (2.4) MPP,BMP, DBCTL AND UTILITIES.......N/A
                         (INC SYNC FAILURE)
(3) NO. OF IFP TRANSACTIONS INCLUDED IN THE SUMMARY OF
              EXCEPTION DETAIL BY TRANSACTION (2.1)+(2.2)...................157837
(4) NO. OF TRANSACTIONS OR PSBS INCLUDED IN THE PROFILE SUMMARY
              FOR ALL REGIONS (INC SYNC FAILURE) BY PSB OR TRANCODE.........157837
(5) NO. OF IFP TRANSACTIONS INCLUDED IN THE OVERALL SUMMARY
              BY TRANSACTION (1)-(2.3).....................................157837
(6) NO. OF TIMES COMBINING CONSTANT WAS DOUBLED.................................0
(7) NO. OF TIMES ENTIRE CI LOGGED (LGNR EXCEEDED)..............................0
```

The meanings of the headings are as follows:

Line (1)
Number of transactions in the analysis period that were examined and selected as a basis for the statistical data reported by the utility. These include any transactions that were involved with Fast Path resources, that is, from IFP, MPP, or BMP regions, or from DBCTL transactions. These are also the transactions written to the total traffic output data set if the SYSUT1 DD statement was provided.

Line (2)
Number of exceptional transactions found and written to the SYSUT2 data set. These include:

- IFP transactions with a transit time equal or greater than the Exceptional Transit Time Specification
- All IFP transactions with a sync point failure
- All IFP transactions for which no dequeue records were found
- All non-message-driven Fast Path transactions if the option NON-MESSAGE was selected by the user. These include MPP, BMP, utility, and DBCTL transactions.

Line (2.1)
Number of IFP transactions with a transit time equal or greater than the Exceptional Transit Time Specification. The number must match the number of transactions reported in the column NO. OF TRANS of the Summary-of-Exception-Detail-by-Transaction-Code-for-IFP report.

Line (2.2)
Number of IFP transactions with a synchronization point failure. The number must match the number of transactions reported in the column SYNC FAIL of the Summary of Exception Detail by Transaction Code for IFP Regions.

Line (2.3)
Number of IFP transactions in the analysis period for which dequeue records were not found.

Line (2.4)
Number of non-message-driven Fast Path transactions. These include all transactions from MPP, BMP and utility regions, and from DBCTL threads found in the analysis period. This is reported only if the NON-MESSAGE option was selected.

If the NON-MESSAGE option is not selected, the N/A (not applicable) characters are printed.

Line (3)
Number of IFP transactions as reported by the Summary of Exception Detail by Transaction Code for IFP Regions. The number includes successfully processed transactions and transactions with a synchronization point failure. It is the sum of the numbers reported in lines (2.1) and (2.2). It does not include transactions for which no dequeue records were received.

Line (4)
Number of transactions included in the Overall Summary of Resource Usage and Contentions for All Transaction Codes and PSBs report. The number must match the number in line (1).

Line (5)
Number of transactions included in the Overall Summary of Transit Times by Transaction Code for IFP Regions. The number must match the number of transactions reported in the NO. OF TRANS column.

Line (6)
Total number of times the LGNR specification was exceeded for all transaction codes.

Line (7)
Total number of times the entire CI was logged for all transaction codes.

Chapter 16. File Select and Formatting Print utility (DFSERA10)

Use the File Select and Formatting Print utility (DFSERA10) to assist in the examination and display of data from the IMS log data set.

The utility can:

- Print or copy an entire log data set
- Print or copy from multiple log data sets based upon control statement input
- Print Operations Manager (OM) log records
- Select and print log records on the basis of sequential position in the data set
- Select and print external trace data sets
- Select and print log records based upon data contained within the record itself, such as the contents of a time, date, or identification field
- Allow modules to special process any selected log records

Use a series of control statements to define the input and output options, selection ranges, and various field and record selection criteria.

Subsections:

- "Restrictions" on page 423
- "Prerequisites" on page 423
- "Requirements" on page 423
- "Recommendations" on page 423
- "Input and output" on page 423
- "JCL specifications" on page 424
- "Utility control statements" on page 425

Restrictions

Currently, no restrictions are documented for the DFSERA10 utility.

Prerequisites

Currently, no prerequisites are documented for the DFSERA10 utility.

Requirements

Currently, no requirements are documented for the DFSERA10 utility.

Recommendations

Currently, no recommendations are documented for the DFSERA10 utility.

Input and output

© Copyright IBM Corp. 1974, 2024

All data input is processed using QSAM and can reside on either tape or direct-access storage devices. Data set organization must be physical sequential. The record format can be fixed or variable in length, blocked or unblocked, or of undefined length. You can use multiple input and output data sets, and they can reside on different device types.

The data set containing control information must have a record length of 80. These statements are reproduced on the output print data set in the same format and sequence as they are processed. If error conditions are encountered, error messages are produced following the statement to which they apply.

Output data can be formatted and printed on the SYSPRINT data set, copied to a specified data set unchanged, or both.

Data to be printed is formatted into 32-byte segments and displayed in both hexadecimal and EBCDIC forms, with the hexadecimal relative offset value preceding each segment.

The flow of control for the program passes through two major stages:

- Control statement processing, where construction of record test and selection parameters takes place and control statement errors are diagnosed
- Record selection and output processing, where the input data is read, analyzed, and compared with the selection parameters to determine the applicability of the record for output

The first phase reads and examines the parameter statements and constructs the required test or test series to create a test group. This test group is then used in record selection when control passes to the next phase of the program. The second phase reads the input data and determines the disposition by the results of each test in the group. When the end of the input data is reached, either by encountering an end-of-file condition or the satisfying the indicated record count, program control shifts back to phase one, where the next group of tests is constructed.

JCL specifications

The File Select and Formatting Print utility executes as a standard operating system job. You must define a JOB statement, an EXEC statement, and DD statements defining input and output.

EXEC statement

Must be in the format

```
// EXEC PGM=DFSERA10
```

Alternatively, the EXEC statement can be included in a cataloged procedure.

DD statements

STEPLIB DD
Defines a partitioned data set containing the EXIT routine modules. If EXIT routines are not used or if the modules reside in LINKLIB, this statement is not required.

SYSPRINT DD
Describes the output data set to contain the formatted print records and control messages. It is usually defined as SYSOUT=A.

The RECFM=FBA DCB parameter is specified for this data set. LRECL and block size can be provided on the SYSPRINT DD statement and must be a multiple of 133. The default for both is 133.

Attention: If you specify a non-default LRECL and the EXITR option statement, you must specify a EXITR module that supports the non-default LRECL. If the EXITR module does not support the non-default LRECL, the default value of 133 must be used.

SYSIN DD
Describes the input control data set. This file must contain fixed-length 80-character records.

input or data DD
Defines the input data set to be examined to produce the formatted print records.

These data sets must be standard labeled files, either direct-access or tape. They can be of any record format (F, FB, V, VB, VBS, or U), as long as they are of DSORG=PS.

If a file with RECFM=U is used, the DCB BLKSIZE parameter must be specified. These files are processed using QSAM. Any file that QSAM supports can be described as input.

If a ddname is not specified in the CONTROL statement, the default ddname used is SYSUT1.

output or data DD
Defines the optional output data set to contain the selected records.

DFSERA10 sets the RECFM of this data set equal to the RECFM specified for the input data set. This is also done for LRECL and BLKSIZE if not specified.

The default ddname used is SYSUT4.

Utility control statements

This utility uses three types of control statements. You can use an additional statement type to provide titles or comments on the output listings. Keyword operands on these statements can be extended to additional statements, to a maximum of 9, by placing a nonblank character in position 72 and continuing the parameter in position 16 of the next statement. Each full keyword has an abbreviation that you can use.

The CONTROL statement defines the ddnames used for the input and output data sets and the beginning and ending limits of the data set being scanned. This statement is optional if the default parameter values are satisfactory.

The OPTION statement defines the test or series of tests performed on the data of the candidate record to determine its qualification for selection. You can execute one or more tests on each logical record by the appropriate number of OPTION statements, creating the logical "OR" function. You can analyze records with the logical "AND" function by creating a test series using the multifield test capability of the COND parameter and the necessary number of OPTION statements. Use the operands COND=M and COND=E to denote the beginning and ending, respectively, of a series for multifield testing of a record.

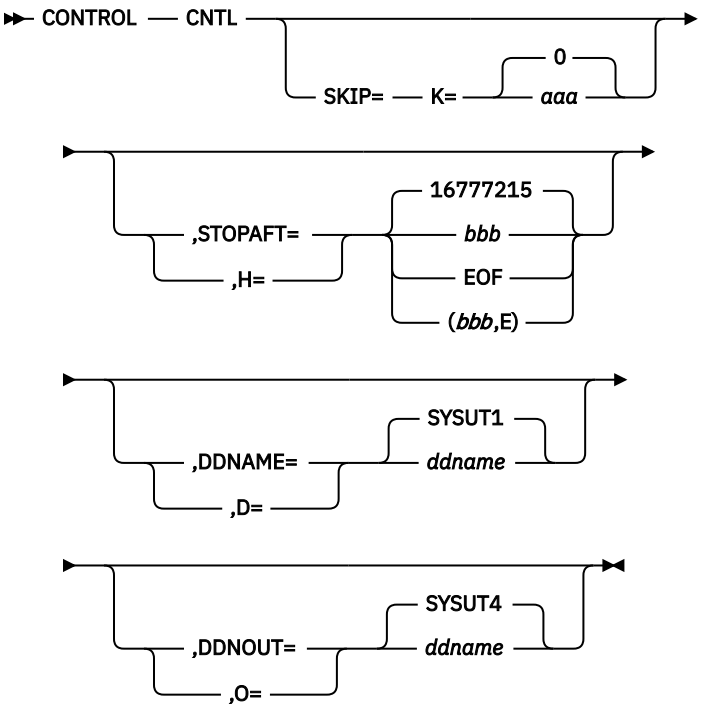
Each OPTION function has its own output processing defaults. If you use multiple OPTION functions to create a multifield test series, final output processing is determined by the OPTION statement coded with the COND=E keyword.

Use the END statement as a delimiter to separate one group of tests (made up of one or more OPTION statements) from subsequent groups of tests on the next data set. When an END statement is encountered in the control input stream, the construction of record selection parameters ceases and the processing of input data records starts. Proper use of the END statement allows one execution of the utility program to perform a varied number of tests on one or more IMS log data sets.

You can use the * or COMMENT statement to include any additional information in identifying tests or data. Those statements have no effect on the utility program.

CONTROL statement

The CONTROL statement is optional. If it is not specified, the SYSUT1 input file is examined. The optional output data set defined on the SYSUT4 DD statement is opened only if you specify the OPTION COPY function in the current group of tests. This data set is used only if COND=E is also specified.



CONTROL or CNTL
Identifies the CONTROL statement.

SKIP= or K=
Defines the first record tested. All prior records are ignored.

If this keyword is not specified, a default value of zero is used and the first record on the input file is tested.

aaa
Must be specified in the range of zero to 99999999, and cannot have embedded commas.

STOPAFT= or H=
Defines the last record to be tested. The current group of tests terminates when this value has been reached by counting processed records.

If this keyword is not specified, a default value of 16777215 is used.

If the STOPAFT parameter uses the default value of 16777215 and message DFS707I indicating EOF does not appear, the records after 16777215 have not been processed.

bbb
Must be specified in the range of 1 to 99999999, with no embedded commas. If the value zero is specified, one record is processed.

EOF
Denotes end-of-file condition. Use of the EOF parameter allows record processing beyond the stated maximum of 99999999 records.

E
Causes records to be counted for test sequence termination only if they satisfy selection criteria. Otherwise, all records read (after the SKIP value) are counted.

DDNAME= or D=
Identifies the input data set for the current group of tests. A corresponding DD statement must be supplied.

If this keyword is not specified, a default of SYSUT1 is used and the appropriate DD statement must be supplied.

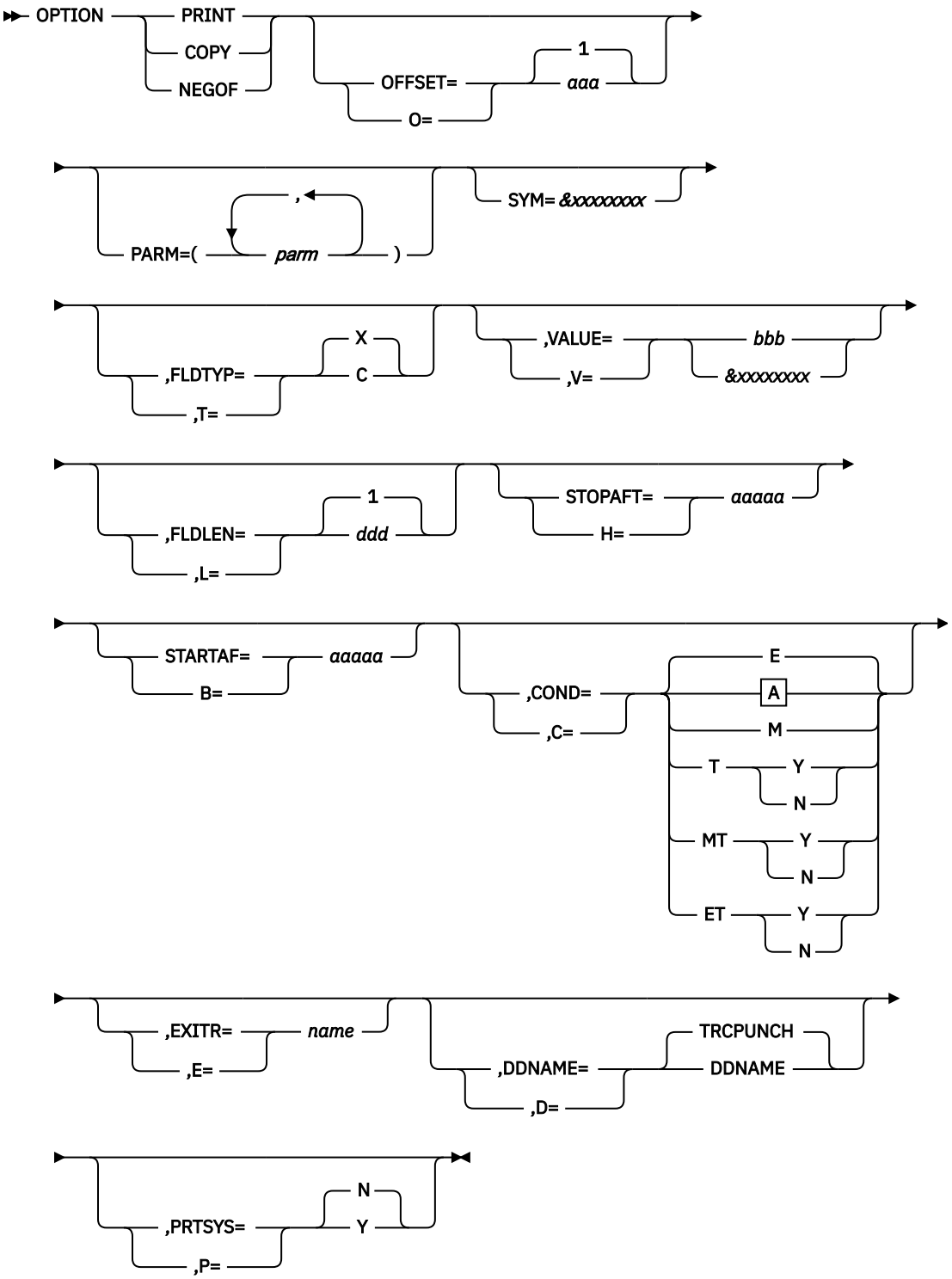
DDNOUT= or O=

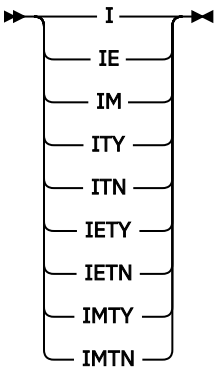
Identifies the optional output data set for the current group of tests.

This keyword is used with the OPTION COPY function and is only required if a ddname other than the default of SYSUT4 is required. (DDNOUT or the presence of SYSUT4 will not cause this data set to be used; this data set will be used only if OPTION COPY is specified with COND=E.)

OPTION statement

The OPTION statement constructs one set of tests. One or more OPTION functions can be specified in any combination desired to further define the selection criteria and output processing performed against each input record. Except for EXITR and DDNAME keyword operands, omitting the keyword operands causes all records processed by phase 2 of this program to be displayed on the SYSPRINT data set or transferred to the specified output data set.


OPTION
PRINT
COPY
NEGOF
OFFSET=
O=
1
aaa
PARM=(
,
parm
)
SYM= &xxxxxxxx
,FLDTYP=
,T=
X
C
,VALUE=
,V=
bbb
&xxxxxxxx
,FLDLEN=
,L=
1
ddd
STOPAFT=
H=
aaaaa
STARTAF=
B=
aaaaa
,COND=
,C=
E
A
M
T
Y
N
MT
Y
N
ET
Y
N
,EXITR=
,E=
name
,DDNAME=
,D=
TRCPUNCH
DDNAME
,PRTSYS=
,P=
N
Y
A



Options

Each option has two distinct functions:

1. Determines starting position for OFFSET keyword
2. Determines output processing to be performed

If individual options are combined to form a multifield test, the use of OFFSET remains unchanged; however, output processing is determined by the OPTION coded with the COND=E keyword.

PRINT
Causes all selected records to be displayed on the SYSPRINT data set.

COPY
Causes all selected records to be transferred to the specified output data set. These records can also be displayed on the SYSPRINT data set by use of the PRTSYS keyword.

NEGOF
Causes the OFFSET keyword value to be used as a negative offset from the end of the log record. All records selected using this function are displayed on the SYSPRINT data set.

Keywords

The following keywords are all optional:

OFFSET= or O=
Is used to define the location of the first byte of the field to be tested in the record. The default is position one of the record.

aaa
Must be specified in the range from 1 up to and including the length of the record under test. Maximum value is 32767 bytes. No checking is performed to determine if the logical record length is exceeded.

If you use DSECTs to locate values in control records or blocks, you must adjust the starting value for the OFFSET parameters. Most DSECTs start with a relative value of ZERO, while the value specified in the OFFSET keyword is always expressed as relative to byte 1.

PARM=
Is used to pass parameters to the DFSERA70 exit routine.

SYM=
Is used to define a value as a symbol. This option replaces the VALUE keyword and must not be used in the same element tests as the VALUE= keyword.

&xxxxxxxx
Is the unique name of a symbol. The '&' is the recognition character. The 'xxxxxxxx' is a 1- to 8-character symbol name. It must be unique for each SYM= specified. This symbol can be used for a VALUE= option in one or many of the following elements in a test series.

FLDTYP= or T=
Is used to define the type of data in the VALUE=field.

X
Defines the data to be treated as hexadecimal pairs. The test data is packed (2 bytes into one to form hexadecimal equivalents). This is the default value.

Example: If VALUE=D9D6D6E3E2C5C7 (14 bytes) is specified with the FLDTYP=X parameter, the resultant VALUE= is: ROOTSET in EBCDIC or D9D6D6E3E2C5C7 in hexadecimal; in either case, the length is only 7 bytes.

C
Defines the data to be treated as EBCDIC. The test data is used as punched in the card, with no alterations.

VALUE= or V=
Defines the characters of the test field. If FLDTYPE=X is specified, this data must be entered as hexadecimal character pairs. For a "test under mask" condition, a single pair must represent the hexadecimal value for the test. If FLDTYP=C is specified, this data must be entered as EBCDIC characters. If the character of blank or comma is to be included in this parameter, FLDTYP=X must be used with the appropriate hexadecimal equivalent.

Restriction: This option must not be used in the same element test as the SYM= keyword.

bbb
Cannot exceed 255 EBCDIC or 510 hexadecimal characters. The length of this field is determined by the FLDLEN= keyword value and not by the number of "nonnull" characters in this field.

&xxxxxxxx
Is the symbol name of a preceding SYM= option. Each symbol has a value associated with it that is determined by the SYM= option.

FLDLEN= or L=
Defines the number of characters to be used from the test field.

ddd
Represents the actual number of bytes to be used, not the number of characters specified in the VALUE= keyword. The acceptable range of values for this field is 1 to and including 255. The default is 1.

STOPAFT= or H=
Defines the number of records to be selected for a single test or a multifield test. This statement can only be specified on the COND=E control statement for each element test.

aaaaa
Can be from 0 to 32767 elements.

STARTAF= or B=
Defines the number of selected records to be skipped for a single or a multifield test. This statement can only be specified on the COND=E control statement for each element test.

aaaaa
Can be from 0 to 32767 elements.

COND= or C=
Defines the type of test and its relationship to other tests in the group. If this keyword is not specified, the default is COND=E.

E
Marks the last (or only) element in a test series. Any OPTION control statements appearing after this form a new series of tests. This allows various tests to be performed on each record and each test series can be used on different fields within the record. Final output processing is determined by the OPTION function defined with this keyword value.

I
Tests the VALUE= value. The record passes if the test fails. This option can stand alone or precede the E, M, or T parameters.

M
Indicates that this is a multifield test. All tests in this series must be satisfied before final output selection and processing of this record begins.

T
Causes the VALUE= byte to be used as a "test under mask" value, instead of a compare field. Only the first byte (two hexadecimal characters if FLDTYP=X) of the VALUE= field is used. If FLDTYP=C is used, the hexadecimal equivalent of the EBCDIC character is the test value. If this parameter is used, the FLDLEN= keyword must not be specified and a default length of 1 is assumed.

Y
Indicates that, for the "test under mask" to be considered satisfied, there must be a bit in the record test field for each corresponding bit of the test byte. This is equivalent to a "branch if ones" test.

N
Indicates that, for the "test under mask" to be considered satisfied, there must not be a bit in the record test field for any of the corresponding bits of the test byte. This is equivalent to a "branch if zeros" test.

MT
Defines a "test under mask" OPTION but with the properties of a multifield test as described in the M parameter. Because the T parameter assumes a default value of 1,e, the MT parameter must be used for a multifield test that starts with a "test under mask" value.

ET
Indicates that a multifield test series ends with "test under mask" condition.

EXITR= or E=
Specifies the entry point name of an exit routine to be given control when a candidate record has satisfied all selection criteria for the current test.

If multiple test groups have specified the same exit routine, an attempt is made to load the routine into storage for each group; therefore, the routine should be re-enterable. Upon reaching end of file on input, a final call is made to the exit routine. You can determine if end of file was reached by checking for zeros in the parameter field.

Interface to the exit routine is as follows:

- ENTRY:

```
REGISTERS
```

R1
Contains a pointer to a parameter list.

R13
Points to an empty save area.

R14
Contains a return address.

R15
Contains the exit routine entry address.

```
PARMLIST
```

- The parameter list consists of two words, the first is a pointer to the candidate record; the second (with the high order bit on) is a pointer to the SYSPRINT data set DCB.

  EXIT:
  Upon return from the exit routine, register 15 is used to determine whether or not processing is to continue on this record.
  A nonzero value indicates that no further processing is done on this record, and selection tests start again against the next input record.

A zero value indicates that this record is required, and output processing is now determined based upon the last OPTION statement encountered containing the COND=E keyword.

If the EXITR keyword is omitted, processing continues as though a return code value of zero had been received.

DDNAME= or D=

Defines the output data set used by the DL/I call trace log record retrieval routine (DFSERA50) whenever it is specified as the user exit routine. A corresponding DD statement must be supplied.

If this keyword is not specified and DFSERA50 is the exit routine, a default of TRCPUNCH is used and the appropriate DD statement must be supplied.

PRTSYS= or P=

Is used to display selected records on the SYSPRINT data set.

N

Indicates that no printing of selected records is done.

Y

Indicates that all records transferred to the output data set are also formatted and printed.

This keyword can only be used with OPTION COPY function. N is the default.

END statement

When you have defined all tests for the current input file, use the END statement to execute those tests.

END is entered at position 1. Positions 10 and on can be used for comments.

COMMENT statement

The COMMENT statement is optional. If used, the contents are displayed on the SYSPRINT data set.

Alternatively, you can use an asterisk (*) in position 1 to indicate a comment; however, comments inserted using an asterisk are not displayed in the SYSPRINT data set.

Related reference

Printing Repository Server audit log records (Diagnosis)

Examples of the DFSERA10 utility

The following examples illustrate some of the ways you can use DFSERA10. Most of the examples refer to the IMS log data set; however, you can use this utility with any data set that can be processed using QSAM.

For clarity, all option keywords are specified in full form, and many are coded where the default could be taken. Use of the short form and keyword defaults greatly reduces the input required.

Example 1

This example shows the JCL and control statements required to print or copy all log records from an IMS log data set.

```
//EXAMPLE1 JOB
//*
//        EXEC PGM=DFSERA10
//STEPLIB   DD DISP=SHR,DSN=IMS.SDFSRESL
//*
//SYSUT1    DD DISP=(OLD,KEEP),DSN=IMSLOG,
//             UNIT=TAPE,VOL=SER=123456
//SYSUT4    DD DISP=(NEW,PASS),DSN=EXAMPLE1.COPY1,
//             UNIT=SYSDA,SPACE=(CYL,(3,1)),
//             VOL=SER=IMSPAC
//*
//SYSPRINT  DD SYSOUT=A
```

```
//*
//SYSIN     DD *
*-------------------------------------------------------------------*
*     CONTROL STATEMENT : DEFAULTS                                  *
*                  INPUT = SYSUT1                                   *
*                 OUTPUT = SYSPRINT                                 *
*  SELECTION QUALIFIERS :                                           *
*            1. DEFAULT = ALL INPUT RECORDS                         *
*-------------------------------------------------------------------*
OPTION   PRINT
END
*-------------------------------------------------------------------*
*     CONTROL STATEMENT : DEFAULTS                                  *
*                  INPUT = SYSUT1                                   *
*                 OUTPUT = SYSUT4                                   *
*  SELECTION QUALIFIERS :                                           *
*            1. DEFAULT = ALL INPUT RECORDS                         *
*-------------------------------------------------------------------*
OPTION   COPY
END
/*
//
```

Example 2

This example shows two ways of selecting and printing all log records of a specific type:

- Specifying one selection qualifier: `TYPE X'16' IN 5TH BYTE = (ALL /SIGN ON/OFF)`
- Specifying two selection qualifiers: `TYPE X'16' IN 5TH BYTE = (LOG RECORD TYPE)` and `FLAG X'01' IN 6TH BYTE = 1 (/SIGN ON - ONLY)`

```
//EXAMPLE2 JOB
//*
//        EXEC PGM=DFSERA10
//STEPLIB   DD DISP=SHR,DSN=IMS.SDFSRESL
//*
//SYSPRINT  DD SYSOUT=A
//*
//LOGIN     DD DISP=(OLD,KEEP),DSN=IMSLOG,
//             UNIT=TAPE,VOL=SER=123456
//*
//SYSIN     DD *
*-------------------------------------------------------------------*
*     CONTROL STATEMENT : SPECIFIED                                 *
*                  INPUT = LOGIN                                    *
*                 OUTPUT = SYSPRINT                                 *
*  SELECTION QUALIFIERS :                                           *
*            1. TYPE X'16' IN 5TH BYTE = (ALL /SIGN ON/OFF)       *
*-------------------------------------------------------------------*
CONTROL  CNTL  DDNAME=LOGIN
OPTION   PRINT OFFSET=5,FLDTYP=X,VALUE=16,FLDLEN=1,COND=E
END
*-------------------------------------------------------------------*
*     CONTROL STATEMENT : SPECIFIED                                 *
*                  INPUT = LOGIN                                    *
*                 OUTPUT = SYSPRINT                                 *
*  SELECTION QUALIFIERS :                                           *
*            1. TYPE X'16' IN 5TH BYTE = (LOG RECORD TYPE)        *
*            2. FLAG X'01' IN 6TH BYTE = 1 (/SIGN ON - ONLY)      *
*-------------------------------------------------------------------*
CONTROL  CNTL  DDNAME=LOGIN
OPTION   PRINT OFFSET=5,FLDTYP=X,VALUE=16,FLDLEN=1,COND=M
OPTION   PRINT OFFSET=6,FLDTYP=X,VALUE=01,COND=ETY
END
/*
//
```

Example 3

This example shows how to print or copy two log record types, each containing a field value (USERID) common to both, but residing at different offsets depending upon the record type.

```
//EXAMPLE3 JOB
//*
//        EXEC PGM=DFSERA10
```

```
//STEPLIB   DD DISP=SHR,DSN=IMS.SDFSRESL
//*
//SYSPRINT  DD SYSOUT=A
//*
//LOGIN     DD DISP=(OLD,KEEP),DSN=IMSLOG,
//             UNIT=TAPE,VOL=SER=123456
//LOGOUT    DD DISP=(NEW,PASS),DSN=EXAMPLE3.COPY1,
//             SPACE(CYL,(3,1)),UNIT=SYSDA,
//             VOL=SER=IMSPAC
//*
//SYSIN     DD *
*-------------------------------------------------------------*
*     CONTROL STATEMENT : SPECIFIED                           *
*                  INPUT = LOGIN                              *
*                 OUTPUT = SYSPRINT                           *
*  SELECTION QUALIFIERS :                                     *
*            1. LOG RECORD TYPE X'16'                         *
*               USERID IN 9TH BYTE (FROM BEGINNING OF RECORD) *
*            2. LOG RECORD TYPE X'50'                         *
*               USERID IN 12TH BYTE (FROM END OF RECORD)      *
*-------------------------------------------------------------*
CONTROL  CNTL  DDNAME=LOGIN
OPTION   PRINT OFFSET=5,FLDTYP=X,VALUE=16,FLDLEN=1,COND=M
OPTION   PRINT OFFSET=9,FLDTYP=C,VALUE=USERAAAA,FLDLEN=8,COND=E
OPTION   PRINT OFFSET=5,FLDTYP=X,VALUE=50,FLDLEN=1,COND=M
OPTION   NEGOF OFFSET=12,FLDTYP=C,VALUE=USERAAAA,FLDLEN=8,COND=E
END
*-------------------------------------------------------------*
*-------------------------------------------------------------*
*     CONTROL STATEMENT : SPECIFIED                           *
*                  INPUT = LOGIN                              *
*                 OUTPUT = LOGOUT                             *
*  SELECTION QUALIFIERS :                                     *
*            *  THE SAME AS FOR THE 'PRINT' AND 'NEGOF' OPTIONS  *
*            ABOVE, BUT SINCE THE 'COPY' OPTION DEFINES AN OUTPUT*
*            DATA SET OTHER THAN SYSPRINT, THIS OPTION MUST BE   *
*            CODED WITH THE 'COND=E' KEYWORD.                    *
*-------------------------------------------------------------*
CONTROL  CNTL  DDNAME=LOGIN,DDNOUT=LOGOUT
OPTION   PRINT OFFSET=9,FLDTYP=C,VALUE=USERAAAA,FLDLEN=8,COND=M
OPTION   COPY  OFFSET=5,FLDTYP=X,VALUE=16,FLDLEN=1,COND=E
OPTION   NEGOF OFFSET=12,FLDTYP=C,VALUE=USERAAAA,FLDLEN=8,COND=M
OPTION   COPY  OFFSET=5,FLDTYP=X,VALUE=50,FLDLEN=1,COND=E
END
/*
//
```

Example 4

This example selects all specified log record types, each containing a common userid value, and both print and transfer these records to the specified output data set.

```
//EXAMPLE4 JOB
//*
//        EXEC PGM=DFSERA10
//STEPLIB   DD DSNAME=IMS.SDFSRESL,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//*
//LOGIN     DD DISP=(OLD,KEEP),UNIT=TAPE
//LOGIN     DD DISP=(OLD,KEEP),DSNAME=IMSLOG,
//             UNIT=TAPE,VOL=SER=IMSPAC
//LOGOUT    DD DISP=(NEW,PASS),DSNAME=EXAMPLE4.COPY1,
//             SPACE=(CYL,(3,1)),UNIT=SYSDA,
//             VOL=SER=IMSPAC
//*
//SYSIN     DD *
*-------------------------------------------------------------*
*     CONTROL STATEMENT : SPECIFIED                           *
*                  INPUT = LOGIN                              *
*                 OUTPUT = (SYSPRINT AND LOGOUT)              *
*                  *  SINCE MULTIFIELD TESTS ARE BEING USED,  *
*                  AND CONSIST OF MULTIPLE OPTION FUNCTIONS,  *
*                  FINAL OUTPUT PROCESSING OF THE SELECTED RECORD  *
*                  IS BASED UPON THE 'COPY' OPTION AND 'PRTSYS=Y'  *
*                  KEYWORD BEING CODED WITH 'COND=E'.         *
*  SELECTION QUALIFIERS :                                     *
*             1. USERID = USERBBBB                            *
*             2. LOG RECORD TYPES (X'16',X'50',X'51',X'52')   *
*-------------------------------------------------------------*
```

```
CONTROL  CNTL  DDNAME=LOGIN,DDNOUT=LOGOUT
OPTION   PRINT OFFSET=9,FLDTYP=C,VALUE=USERBBBB,FLDLEN=8,COND=M
OPTION   COPY  PRTSYS=Y,OFFSET=5,FLDTYP=X,VALUE=16,FLDLEN=1,COND=E
OPTION   NEGOF OFFSET=12,FLDTYP=C,VALUE=USERBBBB,FLDLEN=8,COND=M
OPTION   COPY  PRTSYS=Y,OFFSET=5,FLDTYP=X,VALUE=50,FLDLEN=1,COND=E
OPTION   NEGOF OFFSET=12,FLDTYP=C,VALUE=USERBBBB,GLDLEN=8,COND=M
OPTION   COPY  PRTSYS=Y,OFFSET=5,FLDTYP=X,VALUE=51,FLDLEN=1,COND=E
OPTION   NEGOF OFFSET=12,FLDTYP=C,VALUE=USERBBBB,FLDLEN=8,COND=M
OPTION   COPY  PRTSYS=Y,OFFSET=5,FLDTYP=X,VALUE=52,FLDLEN=1,COND=E
END
/*
//
```

Example 5

This example copies selected log records to individual output data sets in one execution of DFSERA10. All selected records are printed.

```
//EXAMPLE5 JOB
//*
//        EXEC PGM=DFSERA10
//STEPLIB   DD DISP=SHR,DSNAME=IMS.SDFSRESL
//*
//SYSUT1    DD DISP=(OLD,KEEP),DSNAME=IMSLOG
//*
//SYSPRINT  DD SYSOUT=A,UNIT=TAPE,VOL=SER=123456
//*
//LOGOUT1   DD DISP=(NEW,PASS),DSNAME=EXAMPLE5.COPY1,
//             SPACE=(CYL,(3,1)),UNIT=SYSDA,
//             VOL=SER=IMSPAC
//LOGOUT2   DD DISP=(NEW,PASS),DSNAME=EXAMPLE5.COPY2,
//             SPACE=(CYL,(3,1)),UNIT=SYSDA,
//             VOL=SER=IMSPAC
//LOGOUT3   DD DISP=(NEW,PASS),DSNAME=EXAMPLE5.COPY3,
//             SPACE=(CYL,(3,1)),UNIT=SYSDA,
//             VOL=SER=IMSPAC
//*
//SYSIN     DD *
*-------------------------------------------------------------------*
*     CONTROL STATEMENT : SPECIFIED                                 *
*                  INPUT = DEFAULT (SYSUT1)                         *
*                 OUTPUT = SYSPRINT AND (LOGOUT1,LOGOUT2,LOGOUT3)   *
*  SELECTION QUALIFIERS :                                           *
*            1. LOG RECORD TYPE X'16'                               *
*            2. USERIDS = (USERAAAA,USERBBBB,USERCCCC)              *
*-------------------------------------------------------------------*
CONTROL  CNTL  DDNOUT=LOGOUT1
OPTION   COPY  OFFSET=9,FLDTYP=C,VALUE=USERAAAA,FLDLEN=8,COND=M
OPTION   COPY  OFFSET=5,FLDTYP=X,VALUE=16,FLDLEN=1,COND=E,PRTSYS=Y
END
*-------------------------------------------------------------------*
CONTROL  CNTL  DDNOUT=LOGOUT2
OPTION   COPY  OFFSET=9,FLDTYP=C,VALUE=USERBBBB,FLDLEN=8,COND=M
OPTION   COPY  OFFSET=5,FLDTYP=X,VALUE=16,FLDLEN=1,COND=E,PRTSYS=Y
END
*-------------------------------------------------------------------*
CONTROL  CNTL  DDNOUT=LOGOUT3
OPTION   COPY  OFFSET=9,FLDTYP=C,VALUE=USERCCCC,FLDLEN=8,COND=M
OPTION   COPY  OFFSET=5,FLDTYP=X,VALUE=16,FLDLEN=1,COND=E,PRTSYS=Y
END
/*
//
```

Example 6

This example shows the JCL and control statements required to print record 158 of an OSAM image copy data set and all type X'50' records on a log data set that refer to this block number (assuming unblocked OSAM).

```
//EXAMPLE6 JOB
//*
//        EXEC PGM=DFSERA10
//STEPLIB   DD DISP=SHR,DSNAME=IMS.SDFSRESL
//*
//SYSUT1    DD DISP=(OLD,KEEP),DSNAME=IMSLOG,
```

```
//            UNIT=TAPE,VOL=SER=123456
//*
//SYSPRINT  DD SYSOUT=A
//*
//IMAGFILE  DD DISP=(OLD,KEEP),DSNAME=OSAMIMAG,
//             UNIT=TAPE,VOL=SER=456789
//*
//SYSIN     DD *
*-------------------------------------------------------------------*
*     CONTROL STATEMENT : SPECIFIED                                 *
*                  INPUT = IMAGFILE                                 *
*                 OUTPUT = SYSPRINT                                 *
*  SELECTION QUALIFIERS :                                           *
*           1. OSAM RBN = 0000009E (RECORD NO. 158)                 *
*-------------------------------------------------------------------*
CONTROL  CNTL  STOPAFT=(1,E),DDNAME=IMAGFILE
OPTION   PRINT OFFSET=1,FLDTYP=X,VALUE=0000009E,FLDLEN=4,COND=4
END
*-------------------------------------------------------------------*
*     CONTROL STATEMENT : DEFAULTS                                  *
*                  INPUT = SYSUT1                                   *
*                 OUTPUT = SYSPRINT                                 *
*  SELECTION QUALIFIERS :                                           *
*           1. LOG RECORD TYPE X'50'                                *
*           2. DATABASE NAME = DATABAS1                             *
*           3. FLAG X'04' IN 7TH BYTE = 0 (OSAM DATA SET)           *
*           4. OSAM RBN = 0000009E                                  *
*-------------------------------------------------------------------*
OPTION  PRINT OFFSET=5,FLDTYP=X,VALUE=50,FLDLEN=1,COND=M
OPTION  PRINT OFFSET=53,FLDTYP=C,VALUE=DATABAS1,FLDLEN=8,COND=M
OPTION  PRINT OFFSET=7,FLDTYp=X,VALUE=04,COND=MTN
OPTION  PRINT OFFSET=43,FLDTYP=X,VALUE=0000009E,FLDLEN=4,COND=E
END
/*
//
```

Example 7

This example shows the JCL and control statements required to print all type X'50' records, where the database name (beginning with the 53rd byte) is not equal to DB01DS01, and to print all type X'25' records.

The second set of control statements uses a symbolic keyword to select the database name, beginning with the 9th byte of the first type X'25' record. Using the same symbolic name for the value in the next control statement, all type X'50' records (except the first) that have the same database name are to be printed beginning with the 53rd byte.

```
//EXAMPLE7 JOB
//         EXEC PGM=DFSERA10
//STEPLIB  DD   DISP=SHR,DSNAME=IMS.SDFSRESL
//*
//SYSPRINT DD   SYSOUT=A
//*
//LOGIN    DD   DISP=(OLD,KEEP),DSNAME=IMSLOG,
//              UNIT=TAPE,VOL=SER=123456
//*
//SYSIN    DD   *
*-------------------------------------------------------------*
*   CONTROL STATEMENT : SPECIFIED                             *
*                INPUT : LOGIN                                *
*               OUTPUT : SYSPRINT                             *
* SELECTION QUALIFIERS:                                       *
*           1. LOG RECORD TYPE X'50'                          *
*              ¬= DB01DS01 STARTING IN THE 53rd BYTE          *
*              (DATABASE NAME) PRINT 5 LOG RECORDS            *
*           2. LOG RECORD TYPE X'25'                          *
*-------------------------------------------------------------*
CONTROL  CNTL  DDNAME=LOGIN
OPTION   PRINT OFFSET=5,FLDTYP=X,VALUE=50,FLDLEN=1,COND=M
OPTION   PRINT OFFSET=53,FLDTYP=C,VALUE=DB01DS01,FLDLEN=8,STOPAFT=5,   X
               COND=IE
OPTION   PRINT OFFSET=5,FLDTYP=X,VALUE=25,FLDLEN=1,COND=E
END
*-------------------------------------------------------------*
*   CONTROL STATEMENT : SPECIFIED                             *
*                INPUT : LOGIN                                *
*               OUTPUT : SYSPRINT                             *
```

```
* SELECTION QUALIFIERS:                                          *
*           1. LOG RECORD TYPE X'25'                             *
*              DEFINE SYMBOL &DBNAME STARTING IN THE 9th BYTE    *
*              (DATABASE NAME ) & PRINT 1 RECORD                 *
*           2. LOG RECORD TYPE X'50'                             *
*              USE SYMBOL &DBNAME FOR DATABASE NAME STARTING     *
*              IN THE 53rd BYTE & SKIP THE FIRST SELECTED        *
*              RECORD                                            *
*----------------------------------------------------------------*
CONTROL  CNTL  DDNAME=LOGIN
OPTION   PRINT OFFSET=5,FLDTYP=X,VALUE=25,FLDLEN=1,COND=M
OPTION   PRINT OFFSET=9,FLDTYP=C,FLDLEN=8,SYM=&DBNAME,STOPAFT=1,COND=E,
OPTION   PRINT OFFSET=5,FLDTYP=X,VALUE=50,FLDLEN=1,COND=M
OPTION   PRINT OFFSET=53,FLDTYP=C,VALUE=&DBNAME,FLDLEN=8,STARTAF=1,    X
               COND=E,
END
/*
//
```

Example 8

This example shows the JCL and control statements required to print an external trace data set.

```
//EXAMPLE8 JOB MSGLEVEL=(1,1)
//*
//PRTTAB  EXEC PGM=DFSERA10
//STEPLIB   DD DISP=SHR,DSN=IMS.SDFSRESL
//*
//SYSUT1    DD DISP=SHR,DSN=IMS.EXTERNAL.TRACE
//*
//SYSPRINT  DD SYSOUT=A
//*
//SYSIN     DD *
*----------------------------------------------------------------*
*     CONTROL STATEMENT : SPECIFIED                              *
*                  INPUT = SYSUT1                                *
*                 OUTPUT = SYSPRINT                              *
*  SELECTION QUALIFIERS :                                        *
*           1. Log record type X'67FA' in fifth and sixth        *
*              byte = (all trace log records)                    *
*----------------------------------------------------------------*
CONTROL  CNTL  SKIP=0
OPTION   PRINT OFFSET=5,VALUE=67FA,FLDLEN=2,COND=E,EXITR=DFSERA60
END
/*
//
```

Example 9

This example shows the JCL and control statements required to print 67FF records from the IMS log.

```
//EXAMPLE9 JOB
//*
//         EXEC PGM=DFSERA10
//STEPLIB  DD  DISP=SHR,DSN=IMS.SDFSRESL
//*
//SYSPRINT DD  SYSOUT=A
//*
//LOGIN    DD  DISP=SHR,DSN=IMSLOG
//*
//SYSIN    DD  *
CONTROL  CNTL  DD=LOGIN
OPTION   PRINT OFFSET=5,FLDTYP=X,VALUE=67FF,FLDLEN=2,COND=M
OPTION   PRINT OFFSET=29,FLDTYP=X,VALUE=F0F8F3F2,FLDLEN=4,COND=E,EXITR=DFSERA30
END
/*
//
```

Example 10

This example shows the JCL and control statements required to print OM log records.

```
//EXAMPLE10 JOB
//*
```

```
//STEP1     EXEC  PGM=DFSERA10
//STEPLIB   DD    DISP=SHR,DSN=IMS.SDFSRESL
//*
//SYSUT1    DD    DCB=(BLKSIZE=32760),DSN=SYSLOG.OM.AUDIT.TRAIL.LOG,
//                SUBSYS=(LOGR,IXGSEXIT)
//*
//SYSPRINT  DD    SYSOUT=A
//*
//SYSIN     DD
CONTROL    CNTL   STOPAFT=EOF
OPTION     PRINT  EXITR=CSLULALE
END
/*
//
```

DFSERA10 utility modules

The File Select and Formatting Print utility calls several modules that pass return codes to the DFSERA10 utility.

The following topics provide additional information about these modules.

Record Format and Print module (DFSERA30)

Use the Record Format and Print Module (DFSERA30) to format trace and general purpose subrecord types (X'00' and X'01') and SNAP subrecord types (X'FD' and X'FF').

Other log records are formatted in z/OS dump format. DFSERA30 is an exit routine of the File Select and Formatting Print Utility (DFSERA10). Because this routine formats log records, it passes a return code to DFSERA10. This return code tells DFSERA10 that the log record has been processed and requires no additional processing.

For trace and SNAP subrecord types, the module creates log record leader information, followed by a formatted printout of each element within the log record.

DFSERA30 translates the STCK value in each record that is dumped into a human-readable date and time stamp, and prints this value on the record header line. Because this value is derived from the hardware clock, you should be aware of the following:

- The time is in UTC (GMT), not local time.
- The hardware clock does not include any leap second adjustments that may be present on your system (see CVT field CVTLSO). Thus, the time printed by DFSERA30 might be different from the time reported by z/OS when the record was written. The difference is equal to the leap second adjustment amount.

Utility control statements

The following figure shows the control statements required to format type X'67' log records using the DFSERA30 exit routine.

```
Column 1     Column 10                                                  72

CONTROL      CNTL
OPTION       PRINT OFFSET=5,FLDLEN=2,VALUE=67aa,                        X
                   COND=E,EXITR=DFSERA30                                X
END
/*
//
```

In this figure, aa is the log record subtype.

aa=01
Specifies TRACE log record subtype

aa=FD
Specifies SNAP log record subtype

aa=FF
Specifies ABEND log record subtype

The following figure shows a sample DFSERA30 output. AE9004 is the storage address of the LXB at the time the log record was created. The second column of each line is the relative offset from the LXB.

```
DFSERA30 -- FORMATTED LOG PRINT
⋮
INTERNAL TRACE RECORD
⋮
LXB
 AE9004   000000    807F0BC9  00093660    00AE9350  00AE92B0    00091E90  00AE991C   17000000  7F0C0000
 AE9024   000020    80000000  520821CE    0008229C  000820C6    80082194  012141CE   60000054  0A000000
 AE9044   000040    30000005  022140C6    600000CE  09000000    30000005  47000000   20000001  00000000
 AE9064   000060    00000000  00000000    00000000  00000000    00000000  00000000   00000000  00000000
 AE9084   000080  TO AE90C4   0000C0    SAME AS ABOVE
 AE90E4   0000E0    00000000  0C419317    F1044193  17F10441    9337E218  D243F510   A314A8C3  419101A2
 AE9104   000100    02F30C41  93179101    A502F004  F30C4193    17F10441  93170000   00000000  00B66218
```

Deadlock report

The deadlock report contains information about the resources and resource owners for deadlocks resulting from 777 and 123 pseudoabends and deadlocks in non-message-driven BMPs. (These result in an 'FD' status code.)

When DFSERA30 encounters this deadlock block, it prints the block and produces a report based on the data in the block. When excessive deadlocks occur, the deadlock block and the report based on it allow an analysis of the resources that are involved in the deadlock.

When IMS abends an application with a 777 or 123 pseudoabend due to an external subsystem detected deadlock, the deadlock report contains information to identify the subsystem, the job, and the unit of recovery that received the deadlock.

Deadlock report for BMP region and MPP region

The following figure is an example of the DFSERA30 report for a simple deadlock involving a BMP region and an MPP region. The MPP program, which is waiting for resource 1 of 2, is chosen as the victim. It is requesting a root lock for key 'KK360'. The BMP program is the holder of this lock. The BMP program is requesting a root lock for key 'KK130'. The MPP program is the holder of this lock.

```
DEADLOCK ANALYSIS REPORT - LOCK MANAGER IS IRLM


RESOURCE DMB-NAME LOCK-LEN LOCK-NAME     - WAITER FOR THIS RESOURCE IS VICTIM
01 OF 02 DHVNTZ02    08    00000BC4800501D7

KEY IS ROOT KEY OF DATA BASE RECORD ASSOCIATED WITH LOCK
KEY=(KK360)

       IMS-NAME TRAN/JOB PSB-NAME DBD/PCB# PST# RGN  CALL LOCK  LOCKFUNC STATE
WAITER SYS3     NQF1     PMVAPZ12 DLVNTZ02 0002 MPP  GET  GRIDX 30400358  06-P
BLCKER SYS3     DDLKBMP1 PLVAPZ22     0027 0003 BMP  ---- ----- --------  06-P



RESOURCE DMB-NAME LOCK-LEN LOCK-NAME
02 OF 02 DHVNTZ02    08    00000924800501D7

KEY IS ROOT KEY OF DATA BASE RECORD ASSOCIATED WITH LOCK
KEY=(KK130)

       IMS-NAME TRAN/JOB PSB-NAME DBD/PCB# PST# RGN  CALL LOCK  LOCKFUNC STATE
WAITER SYS3     DDLKBMP1 PLVAPZ22 DLVNTZ02 0003 BMP  GET  GRIDX 30400358  06-P
BLCKER SYS3     NQF1     PMVAPZ12     0006 0002 MPP  ---- ----- --------  06-P

DEADLOCK ANALYSIS REPORT - END OF REPORT
```

In the example, X'30' is reported under LOCK as GRIDX. Familiarity and some understanding of DL/I locking terminology and data organizations is needed for a full understanding of the formatted deadlock information provided.

How to read the report

The formatted report is summarized by lock name. It begins with lock 1 of n, showing the database name being locked, the lock name length, and the lock name itself. The lock name is composed of codes that provide information about the lock such as its relative block address (RBA), whether the lock occurred in a full-function (FF) or Fast Path (DEDB) database, and—in the case of a DEDB—whether the lock occurred at the control interval (CI) level or at the segment level.

In an FF database, the RBA is displayed in bytes 1–4 of the lock name. For example, in lock name 00000924800501D7, the RBA= 924.

Determining the RBA of a lock in an FP database is slightly more complex. The following table shows the lock name of an FP database is broken down.

Table 22. Lock name in an FP database

Byte position	Lock information
1	Lock ID
2-4	Relative Byte Address
5-6	DMCB Number
7	Area Number
8	Fast Path ID=C6

In an FP database, the first two digits (Byte 1) display the number "80" if the lock occurred at the segment level. In this case, the next 3 bytes displayed indicate the 30 bit RBA. You must multiply the displayed RBA by 4 to get the true RBA.

The first byte of the lock name is the lock ID which identifies the resource being locked. You can find the DSECT in ADFSMAC(DBFEPST) which gives all the possible lock IDs for Fast Path locks.

```
EPSTRSID DS    0F          RESOURCE NAME
EPSTLKID DC    X'00'       LOCK SUB ID
EPSTCILK EQU   X'00'       CI LOCK
EPSTSLLK EQU   X'80'       SLL LOCK - '80' ON &
EPSTSLLL EQU   X'40'                - '40' OFF
EPSTUWLK EQU   X'E4'       UOW LOCK, X'E4'=C'U'
EPSTDULK EQU   X'F0'       DUMMY LOCK
EPSTMDLK EQU   X'F1'       MSDB LOCK
EPSTOVLK EQU   X'F2'       BUFFER OVERFLOW LOCK
EPSTNRLK EQU   X'F3'       DBRC NOT REGISTERD DB
EPSTARLK EQU   X'F8'       AREA LOCK
EPSTCMLK EQU   X'FF'       COMMAND LOCK
```

If the lock occurred at the CI level, the first two digits indicate the code X'00'. In this case, the next 3 bytes displayed indicate the 24 bit RBA. You must multiply the displayed RBA by 256 (X'100') to get the true RBA.

In addition, for any lock that occurred in an FP database, the last two digits (Byte 8) of the lock name display the code "C6."

For example, the lock name 80000C02800101C6 occurred in an FP database at the segment level with an RBA of '00003008'.

In many cases, the lock is for a database record for which the root key is known. The next lines provide information about the root key for the database record being locked. The following are the possible report statements for the root key.

- `KEY IS ROOT KEY OF DATA BASE RECORD ASSOCIATED WITH LOCK`

  This statement is the most common. It indicates that the key that follows is the root key for the database record involved in the lock. You see this report statement, for example, when a HIDAM or PHIDAM root is retrieved using the index. The key is known when the lock on the root is requested.

- `KEY FOR RESOURCE IS NOT AVAILABLE`

  This statement indicates that the key for the database record being locked is not available. You see this report statement, for example, when a GN call for an HDAM or PHDAM database causes DL/I to lock the next root anchor. When this lock request is one of the resources involved in the deadlock, it is not possible to print the key associated with the lock.
- `LOCKING PRIOR ROOT FOR HIDAM ROOT INSERT, KEY DISPLAYED IS FOR NEXT HIGHER ROOT`

  This statement can occur when a root is inserted in HIDAM or PHIDAM and the root has twin forward and backward pointers. You see this report statement, for example, if the keys 10 and 12 are present and 11 is being inserted. The key displayed is key 12 but the lock is on key 10.
- `LOCKING ON NEXT HIDAM ROOT FOR GN CALL, KEY DISPLAYED IS FOR PRIOR HIDAM ROOT`

  This statement can occur when using HIDAM or PHIDAM with twin forward and backward pointing, and keys 10, 11, and 12 exist, and position is on key 10; a GN call requires a lock on 11. When the lock is required, the key is not known, so the key of the prior root is displayed.
- `LOCKING ON HDAM ANCHOR, KEY DISPLAYED IS HDAM KEY REQUESTED`

  This statement can occur when using HDAM or PHDAM. The item locked is the anchor. When the anchor is locked, the key that will be retrieved is not known but the key that is requested is known, and it is displayed.

The following report statement can be issued if the locks involve DEDB:

- `SPECIAL INTERNAL LOCK FOR PROMOTE`

  Fast Path requests lock at READ (01) level or at EXCLUSIVE (04) level. This statement can occur when a GU call is followed by a GHU call for the same segment. The resultant request for the lock to be promoted from level 01 - 04 caused a deadlock.

The lock waiter and holder/owner information is printed next. Each waiting and holding work unit is identified by IMSID, tranname or jobname, PSB name, PST number, and region type. The WAITER listed is the work unit that the database key information pertains to.

There are some differences between the two lines of waiter and holder information. The current PCB name, the DL/I call, and the lock request pertain only to the waiter. This information is not available for the holder of the lock. The relative DBPCB number is printed for the holder of the lock.

The current DL/I call being processed is reported as one of the following calls:

GET
DL/I call was GU, GHU, GN, GHN, GNP, or GHNP. The captured information does not allow a breakdown of the specific GET call function.

REPL
DL/I call was REPL.

ISRT
DL/I call was ISRT or ASRT.

DLET
DL/I call was DLET.

POS
DL/I call was POS call on MSDB.

The lock request function is reported under the columns for LOCK and LOCKFUNC. The first byte of the LOCKFUNC is translated for convenience. The LOCKFUNC is the hexadecimal function, mode, state, and flags as mapped by the LRHPARM DSECT.

The reason for translating the lock request function is to identify deadlocks caused by block level data sharing, by application programs accessing data in a different order, or mixtures of both. For deadlock purposes, the lock request functions can be summarized by the following calls, where *x* is usually L, X, B, P, U, or W:

GBID*x*
Get a block lock. Block level sharing only.

GZID*x*
Get a data-set-busy lock. Used only to serialize data set opens, closes, and extensions. Any involvement in a deadlock is probably an indication of an error in IMS code.

GXID*x*
Get a data-set-extend lock. Used to serialize the extending of a data set. Block level sharing only and probably a HISAM database.

GRID*x*
Get a lock on the root of a database record.

GQCM*x*
Get a Q command code lock. This is an application-originated lock on specific data. The GQCM function applies to full function only (Fast Path does not obtain a new lock when the Q command code is issued).

GSEG*x*
Get a segment lock for a dependent segment. This is not used when IRLM is the lock manager.

GFPLL
Get a Fast Path lock.

Lock states

The lock state is the type or level of lock and is typically designated by a number. To manage the lock states, IMS uses either the internal resource lock manager (IRLM) or the program isolation (PI) lock manager. These two lock managers do not use the same states to reflect the level of the locks. The PI lock manager supports four states and IRLM supports 11, though IMS uses only eight of them.

Sometimes the lock states are referred to with names rather than numbers. The names used for the four PI-supported states are:

State 1
Read only

State 2
Read

State 3
Update

State 4
Exclusive

The following table is a matrix that describes the compatibility of the level of an incoming lock request and the level that the lock is held at when using the PI lock manager. A compatible state is indicated by a "C" (meaning that the lock request will be granted) and an incompatible state by an "X" (meaning that the lock request will not be granted).

Table 23. PI lock compatibility matrix

Requested Level	1	2	3	4
Held at 1	C	C	C	X
Held at 2	C	C	X	X
Held at 3	C	X	X	X
Held at 4	X	X	X	X

The eight states provided by the IRLM and their characteristics are defined in two matrices. One is used to determine *resultant state* and the other to determine compatibility for a requesting and holding work unit.

The concept of a *resultant state* requires some explanation. In simple terms, the resultant state is the lock state that results from granting the current request or the "held at" state that a subsequent requestor will see assuming that the current request is granted. Because the IRLM allows a resource to be locked more than once by a given work unit, when a work unit locks a resource for the second time and specifies a different state, the state in which the lock is finally held should be one that carries the privileges of the second state without losing those conferred by the first.

Given a set of states with a strictly hierarchical privilege order, it would be sufficient to grant the higher of the two states. However, to allow a locking protocol in which each higher state does not necessarily include all the privileges of the preceding one, the matrix can specify that the resultant state is a third state that confers the sum of the privileges of the other two states. The request is then processed as a request for the third state. The following table is the resultant state matrix.

Table 24. IRLM resultant state matrix

Requested level	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Held at 1	1	2	3	4	5	6	7	8
Held at 2	2	2	3	4	5	6	7	8
Held at 3	3	3	3	6	5	6	7	8
Held at 4	4	4	6	4	5	6	7	8
Held at 5	5	5	3	5	5	6	7	8
Held at 6	6	6	6	6	6	6	7	8
Held at 7	7	7	7	7	7	7	7	8
Held at 8	8	8	8	8	8	8	8	8

The following table shows the compatibility matrix with compatibility indicated by a "C" and incompatibility by an "X".

Table 25. IRLM compatibility matrix

Requested level	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Held at 1	C	C	C	C	C	C	C	X
Held at 2	C	C	C	C	C	C	X	X
Held at 3	C	C	C	X	X	X	X	X
Held at 4	C	C	X	C	C	X	X	X
Held at 5	C	C	X	C	X	X	X	X
Held at 6	C	C	X	X	X	X	X	X
Held at 7	C	X	X	X	X	X	X	X
Held at 8	X	X	X	X	X	X	X	X

For the IRLM, the state can have an attribute of private. The private attribute is only significant when using block level data sharing. The private attribute has no impact on granting locks to different threads of a single IMS. The private attribute indicates that the lock should be private (only granted) to this IMS.

Restriction: Any thread of another IMS sharing the data cannot be granted the lock.

Private is indicated with a '-P' following the lock state.

Block limits

A fixed-size block is used to hold the data for each resource in the deadlock cycle. This block is large enough to hold the data for a cycle which involves nine resources. If the cycle involves more than nine resources a message indicates this and only the first nine are reported.

There are a limited number of blocks to hold the data. If all the blocks are in use when a deadlock occurs, a message indicates this and no deadlock information is provided for that deadlock.

Additional information gathered

The formatted deadlock report is a summarization of the complete data gathered and snapped to the log. There are two macro DSECTs that map information in the raw data. These are the DIPENTRY DSECT and the DLKDLD DSECT.

Anomaly in deadlock reporting

There is one deadlock situation where the report is different.

```
LOCK 1                  LOCK 2
PST 1 owns STATE SHR    PST 2 owns STATE UPD
PST 3 waits STATE UPD   PST 1 waits STATE UPD
PST 2 waits STATE SHR
```

If application 3 on PST 3 had not interfered by asking for LOCK 1 at an incompatible state, there would have been no deadlock, because PST 2 is asking for LOCK 1 in a compatible state with the owner PST 1.

The anomaly that occurs in the reporting of this deadlock is that LOCK 1 is listed twice. It is listed once with PST 1 owning and PST 3 waiting, and it is listed again with PST 2 waiting and no holder information. The report displays NOTAVAIL in the IMS-NAME field for the BLCKER.

Selecting only the deadlock element from a snap

Fewer elements are snapped on a 777 pseudo abend than for other pseudo abends; however, the snap does include more elements than the deadlock block. It is possible to select only a specific element from a snap. The following figure contains the DFSERA10 control statements to select only the deadlock element from any pseudoabend snap.

```
//SYSIN    DD *
*-------------------------------------------------------------------*
*                                                                   *
*      CONTROL STATEMENT : DEFAULTS                                 *
*                  INPUT = SYSUT1                                   *
*                 OUTPUT = SYSPRINT                                 *
*                                                                   *
*   SELECTION QUALIFIERS :                                          *
*            1. LOG RECORD TYPE OF X'67FF'                          *
*            2. NAME OF BLOCK WITHIN SNAP IS C'DEADLOCK'            *
*                                                                   *
*           EXIT ROUTINE = DFSERA30                                 *
*                                                                   *
*-------------------------------------------------------------------*
OPTION   PRINT  OFFSET=5,FLDLEN=2,VALUE=67FF,FLDTYP=X,COND=M
OPTION   PRINT  OFFSET=33,FLDLEN=8,VALUE=DEADLOCK,FLDTYP=C,COND=E,     X
                EXITR=DFSERA30
END
/*
//
```

IMS-issued subsystem detected deadlocks

When IMS abends an application with U777 because of an external subsystem detected deadlock, the Deadlock Report contains information to identify the subsystem, job, and Unit Of Recovery that received the deadlock.

The following figure is an example of the DFSERA30 report for a deadlock detected by an external subsystem.

```
PSEUDO ABEND RECORD  ABEND NO = 0777  RECNO = 00000162  TIME  12:24:07.1   DATE 2006.292
DEADLOCK

EXTERNAL SUBSYSTEM SSOP     DETECTED A DEADLOCK DURING NORMAL CALL
REGION TYPE   : MPP
REGION NUMBER : 0001
JOB NAME      : MPP1
PSB NAME      : DCSQL7B
SMB NAME      : TXSQL7B
RECOVERY TOKEN: E2E8E2F340404040000000000500000000
```

Related concepts

Locking mechanisms and database integrity in DB/DC and DCCTL (System Administration)

Program Isolation Trace Record Format and Print module (DFSERA40)

The Program Isolation (PI) Trace Format and Print module receives type X'67FA' log records as an exit routine from the File Select and Formatting Print utility (DFSERA10) and formats the records on the SYSPRINT data set.

This topic contains Product-sensitive Programming Interface information.

These log records are produced by the PI (program isolation) trace, trace PI enqueue and dequeue calls to DFSLRH00, and also by DL/I calls to the DL/I analyzer. The DL/I analyzer processes all DL/I calls. When tracing is active, the DL/I analyzer calls are traced. The standard ENQ/DEQ call is invoked by the DFSLR macro instruction.

PI tracing is executed by the **/TRACE** command in an IMS online environment or by the OPTIONS statement with LOCK=OUT specified.

In a data sharing environment, if the PI trace is active and being logged, the PI trace logger is activated by the IMS lock manager (DFSLMGR0) and exits to the IRLM.

The PI Trace Record Format and Print module is loaded during the execution of the File Select and Formatting Print utility and must reside in the LINKLIB or in a JOBLIB or STEPLIB data set.

Output

The following figure is a sample output from DFSERA40. The spacing of fields is altered.

```
DATE:    05/11/10
MODULE  PST   TIME (*=ET)  CALLR   ACT     LEV      WC  WFC   SEQN   FDBK   RC   PC  ID= (RBA   DMB DCB SUF) CLS
TOKEN   COMMENT

LRH0    01                        GZIDB                    0ABE
UNCHN   02                                                 0AC0               61
LRH0    01                        RZIDP                    0AC1
PIEX    01    23:36:22.472   DLI   TNFQ    UPD      00  00    0AC3   0000            481075C5   8007  01

LRH0    01                        TTLKX                    0AC4
PIEX    01    23:36:22.472   DLI   ENQ     UPD      00  00    0AC5   0000            00000658   8006  01
00722050
LRH0    01                        GRIDX                    0AC6
PIEX    01    23:36:22.474   APP   ENQ     SHR      00  00    0ACA   0000            00000694   8006  01      0
007220DC
LRH0    01                        GCCMX                    0ACB
DLA0    01    23:36:22.493         GU                      0ACE
8                                 DL/I CALL
PIEX    01    23:36:22.493   DLI   IDEC    UPD      00  00    0ACF   0000            00000658   8006  01
LRH0    01                        RRIDX                    0AD0
LRH0    01                        GZIDB                    0AC1
LRH0    01                        RZIDB                    0AD4
PIEX    01    23:36:22.495   DLI   ENC     UPD      00  00    0AD6   0000            00001108   8006  01
00722050
LRH0    01                        GRIDX                    0AD7
PIEX    01    23:36:22.496   DLI   IDEQ    UPD      00  00    0ADA   0000            00001108   8006  01
LRH0    01                        RRIDX                    0ADB
```

```
DLA0    03    23:36:23.614         GU                        0ADE
1                             DL/I CALL
LRH0    03                         GZIDB                     0AE4
LRH0    03                         RZIDB                     0AE7
PIEX    03    23:36:23.735   DLI   TENQ    UPD      00  00   0AE9   0000           48107105   8007  01
LRH0    03                         TTLKX                     0AEA
PIEX    03    23:36:23.736   DLI   ENC     UPD      00  00   0AEB   0000           00000408   8006  01
00722050
LRH0    03                         GRIDX                     0AEC
PIEX    03    23:36:23.737   APP   ENQ     SHR      00  00   0AF0   0000           00000428   8006  01      0
00722014
LRH0    03                         GQCMX                     0AF1
DLA0    03    23:36:23.834         GU                        0AFS
2                             DL/I CALL
PIEX    03    23:36:23.835   DLI   IDEQ    UPD      00  00   0AFA   0000           00000408   8006  01
LRH0    03                         FRIDX                     0AFP
LRH0    03                         GZIDB                     0AFC
LRH0    03                         RZIDB                     0AFF
PIEX    03    23:36:23.838   DLI   ENQ     UPD      00  00   0B01   0000           00001108   8006  01
00722050
LRH0    03                         GRIDX                     0B02
PIEX    03    23:36:23.840   DLI   TDEQ    UPD      00  00   0B05   0000           00001108   8006  01
LRH0    03                         RRIDX                     0B06
DLA0    02    23:36:27.257         GHU                       0B0F
4                             DL/I CALL
PIEX    02    23:36:27.257   DLI   TDEQ    UPD      00  00   0B10   0000           0000087C   8006  01
LRH0    02                         RRIDX                     0B11
LRH0    02                         GZIDB                     0B12
LRH0    02                         RZIDB                     0B15
PIEX    02    23:36:27.263   DLI   TENQ    UPD      00  00   0B17   0000           481071C5   8007  01
LRH0    02                         TTLKX                     0B18
PIEX    02    23:36:27.263   DLI   ENQ     UPD      00  00   0B19   0000           00000408   8006  01
007220A0
LRH0    02                         GRIDX                     0B1A
PIEX    02    23:36:27.265   DLI   TENQ    UPD      01  00   0B1E   1800  04       00000428   8006  01
00722014
PIEX    03    23:36:45.079   APP   CEQ     SHR      00  00   0B34   0000                                  0
LRH0    03                         RQCML                     0B35
PIEX    02        0:17.850*  DLI   UNK     RD                0B37
6F                                  SEQ2=0B1E
LRH0    02                         TTLKL                     0B38        04
DLA0    02    23:36:45.982         GHU                       0B3A
5                             DL/I CALL
PIEX    02    23:36:45.982   DLI   TDEQ    UPD      00  00   0B3B   0000           00000408   8006  01
LRH0    02                         RRIDX                     0B3C
LRH0    02                         GZIDB                     0B3D

LRH0    02                         BZIDB                     0B40
PIEX    02    23:36:45.986   DLI   TENQ    UPD      00  00   0B42   0000           481075C5   8007  01
LRH0    02                         TTLKX                     0B43
PIEX    02    23:36:45.986   DLI   ENQ     UPD      00  00   0B44   0000           00000658   8006  01
007220A0
LRH0    02                         GRIDX                     0B45
```

Explanation of column headings

DATE
Specifies the date PI trace started. The TIME field is relative to this date.

MODULE
Specifies the module that issued the DFSLR call to DFSLRH00 or the module that called the IRLM or DFSFXC10. The four characters selected come from the xxxx portion of the full module name DFSxxxx0.

PST
Specifies the program specification table (PST) number (from PSTPSTNR).

TIME
Specifies the time of the call as HHH:MM:SS.UUU, where UUU is milliseconds, relative to the date on which tracing started. If the return code (RC) is 04 and PI trace timing is active at the time of the call, the next record for this PST in this report shows the elapsed time of the enqueue wait in this field. The time is indicated as MM:SS.UUU*, with the "*" indicating it is an elapsed time.

CALLR
Specifies the type of caller (DLI, FP, APP).

ACT
Specifies the action requested.

LEV
Specifies the level of control for this call.

RD
Read only

SH
Share

UPD
Update

EXC
Exclusive

WC
Number of PSTs that hold this resource in a state that caused this PST to wait.

WFC
Number of PSTs waiting for this PST to release this resource.

SEQN
Specifies the sequence number of the corresponding internal trace.

FDBK
Is 2 bytes of feedback information from either DFSFXC10 or the IRLM.

RC
Specifies the return code from DFSFXC10 or the IRLM.

00
Successful completion

04
Caller must IMS wait for control of the requested resource

08
Deadlock; request is disallowed. This transaction causes an internal pseudoabend, a backout, and automatic rescheduling.

0C
Invalid call

PC
Specifies the PST post code following the enqueue wait. This field is only present when RC is 04 and the TIME field has an "*" at the end.

60
Deadlock occurred. This transaction causes an internal pseudoabend, a backout, and automatic rescheduling.

61
PST was removed from the PI wait chain for a PI lock on behalf of a /STO REG# ABDUMP command.

6F
Control of the resource has been obtained.

ID=
Specifies an 8-byte identification of the resource being enqueued or dequeued. It contains a 4-byte RBA, a 2-byte DMB number, a 1-byte DCB number, and a 1-byte SUF (suffix) field.

CLS
For APP types of callers, specifies the Q-command code class requested. For LMGR traces, specifies the CLASS parameter.

CLS applies to full function only (Fast Path does not support lock class).

TOKEN
Is the address of the control block enqueued or locked on this call or, if the type of call is an unlock or DEQ call, the address of the control block being passed to the lock manager.

COMMENT
Specifies 'DL/I CALL' if a trace is requested from DFSDLA00. Other comments are for LMGR traces.

Utility control statements

The following figure shows the control statements required for DFSERA40.

```
Column 1     Column 10     Column 16                                        72

CONTROL      CNTL
OPTION       PRINT         OFFSET=5,VALUE=67FA,FLDLEN=2,                    X
                           COND=E,EXITR=DFSERA40
END
/*
//
```

DL/I Call Image Capture module (DFSERA50)

If trace data is sent to the IMS log data set, you can retrieve it using utility DFSERA10 and special DL/I call image capture routine DFSERA50. DFSERA50 deblocks, formats, and numbers the DL/I Call Image Capture records to be retrieved.

To use DFSERA50, insert a DD statement defining a sequential output data set in the DFSERA10 input stream. The default ddname for this DD statement is TRCPUNCH. The statement must specify BLKSIZE=80. You can distinguish between output from several BMP applications because the first three bytes of the trace entry sequence number are the PST number.

Utility control statements

The following figure shows the control statements for formatting the DL/I call image capture data (in a format acceptable as input for the DL/I test program DFSDDLT0):

```
Column 1     Column 10                                                    72

OPTION       PRINT OFFSET=5,VALUE=5F,COND=M
OPTION       PRINT OFFSET=25,FLDTYP=C,VALUE=psbname,                      X
                   FLDLEN=8,COND=E,EXITR=DFSERA50,DDNAME=OUTDDN
END
/*
//
```

Use the DDNAME= parameter to name the DD statement used by DFSERA50. The data set defined on the OUTDDN DD statement is used instead of the default TRCPUNCH DD statement. For this example, the DD appears as:

```
//OUTDDN DD ...,DCB=(BLKSIZE=80),...
```

IMS Trace Table Record Format and Print module (DFSERA60)

The IMS Trace Table Record Format and Print module (DFSERA60) receives type X'67FA' log records from the File Select and Formatting Print utility (DFSERA10) and formats the records on the SYSPRINT data set. These log records are produced when you use the OPTION statement for the DFSVSAMP data set or DFSVSMnn PROCLIB member to specify that trace table be written to the log.

DFSERA60 is loaded during execution of DFSERA10 and must reside in the LINKLIB or in a JOBLIB or STEPLIB data set.

Utility control statements

The following figure shows the control statements required to invoke DFSERA60.

```
Column 1    Column 10                                                     72

CONTROL     CNTL
OPTION      PRINT OFFSET=5,FLDLEN=2,VALUE=67FA,                           X
                  COND=E,EXITR=DFSERA60
```

```
END
/*
//
```

Enhanced Select module (DFSERA70)

Use the Enhanced Select module (DFSERA70) to: produce expanded log records from compressed IMS logs, select and format '5X' (DL/I 5X and fast path 5950) log records based upon data contained within the record itself, such as the contents of a time, date, or identification field. These records are formatted along with all log record types listed under the PARM TOKEN=description, and change the format of log output to identify and emphasize some optional log fields.

You specify the search criteria for the routine as subparameters of the PARM= parameter of the OPTION statement for the File Select and Formatting Print utility (DFSERA10). The possible subparameters of PARM= are:

XFMT=
Extends the X'50' log record format to enhance the retrievability of certain data entries.

Y
Highlights the log data for certain types of processing by placing the data on a separate line and adding identifiers for data entries. It applies to log data that describes the following types of processing: data sharing, XRF buffer and lock tracking, space management, key, backout (undo), and recovery (redo). If a type of processing is not relevant, the data section is omitted.

These data sections are added after the raw log data for the record. Each section includes identifiers followed by hexadecimal log data, character log data, or both. They contain the following entries, where X represents hexadecimal log data and C represents character log data:

Data sharing

```
DSHRDSSN XXXXXXXX DSHRLSN XXXXXXXXXXXX DSHRUSID
XXXXXXXX RACF-UID CCCCCCCC XXXXXXXXXXXXXXXX
```

XRF buffer and lock tracking

```
TRAKPLSZ XXXX TRAKBUFN XXXX TRAKHASH XXXXXXXX
TRAKLOCK XXXXXXXX TRAKFLGS XX XX
```

Space management

```
SMGTFLGS XX XX SMGTROFF XXXX SMGTRLEN XXXX
```

Key

```
KSDS  Character string describing database action
LENGTH XXXX
One or more lines of mixed hexadecimal and character data
```

Undo

```
UNDO  Character string describing database action
LENGTH XXXX  OFFSET XXXX
One or more lines of mixed hexadecimal and character data
```

Redo

```
REDO  Character string describing database action
LENGTH XXXX  OFFSET XXXX
One or more lines of mixed hexadecimal and character data
```

N
Does not highlight the log data for data sharing, buffer and lock tracking, space management, key, backout or recovery. The data is formatted as part of the raw data for the record.

N is the default.

PST=*pst_number*
Selects records for the PST number.

SYSID=*system_id*
Selects records for the system ID portion of recovery token.

TOKEN=*token*
Selects records for the hexadecimal token portion of recovery token. You can select the following record types: X'07', X'08', X'0A', X'13', X'27', X'28', X'31', X'32', X'35', X'37', X'38', X'39', X'3D', X'41', X'4C', X'50', X'56', X'59', X'5901', X'5903', X'5937', and X'5938'.

PSB=*psb_name*
Selects records for the PSB name.

DBD=*dbd_name*
Selects records for the DBD name.

RBA=*rba_value*
Selects records for the RBA (lrecl).

BLOCK=*block_rba*
Selects records for the RBA (block).

USERID=*userid*
Selects records for the userid.

KEY=*ksds_key*
Selects records for the key.

OFFSET=*offset*
Selects records that update a given offset of data in the buffer.

UNDO=*undo_data*
Selects records for backout data that matches the character string you specify. The maximum length of the character string is 255 characters.

REDO=*redo_data*
Selects records with recovery data that matches the character string you specify. The maximum length of the character string is 255 characters.

DATA=*log_data*
Selects records with data, including compressed data, anywhere in the record that matches (searches all log records). The maximum length of the character string is 255 characters.

Each subparameter must be uppercase and not have any blanks. The subparameter data must be character or decimal. Hexadecimal data must be preceded by an X and the data enclosed in single quotes (for example, X'0123').

Once the record is selected, it can be written to tape or DASD.

When multiple subparameters are specified, all conditions must be met to select a record. Use multiple routines to select records if some of the conditions have been met.

The log print formatting is done by DFSERA30. The format appears as if DFSERA30 was the routine specified. DFSERA30 must be available for DFSERA70 to load.

Unrecognized characters or invalid parameter specifications are ignored by this routine.

Examples

These examples show how to use the DFSERA70 module to print records with regular or expanded data.

The following example shows the option for printing all records that include X'50' or X'5950' database records and expanding the data in the X'5050' records.

```
OPTION  PRINT EXITR=DFSERA70
```

The following example shows the option for printing only X'50' database records with expanded data.

```
OPTION  PRINT O=5,V=50,EXITR=DFSERA70
```

The following example shows the option for printing X'50' database records with expanded data and 67 diagnostic records.

```
OPTION  PRINT O=5,V=67,EXITR=DFSERA30
OPTION  PRINT O=5,V=50,EXITR=DFSERA70
```

The following example shows the option for printing all records in regular format including X'50' or X'5950' database records for a PST number of X'A' using a PSB named APPLPSB.

```
OPTION  PRINT EXITR=DFSERA70,PARM=(XFMT=N,PST=X'A',PSB=APPLPSB)
```

The following example shows the option for printing all records in regular format including X'50' or X'5950' database records at an RBA of X'2000' and an offset of X'200'.

```
OPTION  PRINT EXITR=DFSERA70,PARM=(XFMT=N,RBA=X'2000',OFFSET=X'200')
```

The following example shows the option for printing, in extended format, all records that contain the character string 'aaaa'.

```
OPTION  PRINT EXITR=DFSERA70,PARM=(XFMT=Y,DATA=aaaa)
```

The following example shows the option for selecting all types of log records with the token X'0001F8FF00000000' and printing the records in extended format.

```
OPTION  PRINT EXITR=DFSERA70,PARM=(XFMT=Y,TOKEN=X'0001F8FF00000000')
```

The following example shows the option for selecting X'0A' log records with the token X'0001F8E400000001' and printing the records in regular format.

```
OPTION  PRINT O=5,V=0A,T=X,EXITR=DFSERA70,PARM=(XFMT=N,TOKEN=X'0001F8E400000001')
```

OM Audit Trail Format and Print module (CSLULALE)

To print the OM log records from the MVS™ system logger, use the IMS File Select and Formatting Print utility (DFSERA10) with module CSLULALE.

JCL specifications

The following example shows the required JCL to print the log records from an MVS system log. This JCL causes the MVS logger to invoke the default log stream subsystem exit routine, IXGSEXIT, to copy the log records. The exit routine returns a maximum of 32760 bytes of data for each log record even though OM supports larger log records. You can specify the name of a different exit routine, if necessary.

Use the following JCL to print the OM log records

```
//CSLERA10 JOB   MSGLEVEL=1,MSGCLASS=A,CLASS=K
//STEP1    EXEC  PGM=DFSERA10
//STEPLIB  DD    DISP=SHR,DSN=IMS.SDFSRESL
//SYSPRINT DD    SYSOUT=A
//SYSUT1   DD    DSN=SYSLOG.OM.AUDIT.TRAIL.LOG,
//               SUBSYS=(LOGR,IXGSEXIT),
//               DCB=(BLKSIZE=32760)
//SYSIN    DD  *
CONTROL    CNTL H=EOF
OPTION     PRINT EXITR=CSLULALE,PARM=(F=option)
```

```
END
//
```

DD statements

STEPLIB
DSN= points to IMS.SDFSRESL, which contains the IMS File Select and Formatting Print utility, DFSERA10.

SYSPRINT
Describes the output data set to contain the formatted print records and control messages. It is usually defined as SYSOUT=A.

The RECFM=FBA DCB parameter is specified for this data set. LRECL and block size can be provided on the SYSPRINT DD statement and must be a multiple of 133. Any LRECL value between 133 and 32 760 can be specified. The default for block size and LRECL is 133.

SYSUT1
DSN= points to the OM log stream name. Use the same name that was specified in the OM startup parameters, in the AUDITLOG= parameter in the CSLOIxxx PROCLIB member.

Limiting log data to a specified time range

You can limit the log records you print to those in a particular interval of time using the FROM and TO parameters on the SUBSYS statement. For example, see the following DD card:

```
//SYSUT1 DD DSN=SYSLOG.OM.AUDIT.TRAIL.LOG,
//    SUBSYS=(LOGR,IXGSEXIT,
//    'FROM=(2010/042,11:00:00),TO=(2010/042,12:00:00)'),
//    DCB=(BLKSIZE=32760)
```

The DD card passes only log records from 11:00 to 12:00 on day 42 of the year 2010 to the DFSERA10 program. Dates and times specified are in GMT (Greenwich Mean Time). The seconds field of the time values is optional.

To use local dates and times, add the LOCAL keyword to the statement, see the following DD card:

```
//SYSUT1 DD DSN=SYSLOG.OM.AUDIT.TRAIL.LOG,
//     SUBSYS=(LOGR,IXGSEXIT,
//     'FROM=(2010/042,11:00:00),TO=(2010/042,12:00:00),LOCAL'),
//     DCB=(BLKSIZE=32760)
```

Utility control statements

PARM=
Is used to pass parameters to the CSLULALE exit routine.

F=
Specifies the parameter name. Valid options include:

WRAP
Specifies wrapping of individual lines. WRAP is the default.

BYCOL
Specifies grouping lines by column.

BYRSC
Specifies grouping lines by resource

H=
Specifies the number of log records to print. H=EOF prints all log records.

Related reference
z/OS: IXGSEXIT - Log Stream Subsystem exit

Chapter 17. IMS Monitor Report Print utility (DFSUTR20)

Use the IMS Monitor Report Print utility (DFSUTR20) to take the data collected by the IMS Monitor (DFSMNTR0) and print summary reports and distribution displays of the data.

The report formats and the nature of information in the reports are identical or similar to those printed by the IMS DB Monitor Print utility (DFSUTR30).

Subsections:

- "Restrictions" on page 455
- "Prerequisites" on page 455
- "Requirements" on page 455
- "Recommendations" on page 455
- "Input and output" on page 455
- "JCL specifications" on page 456

Restrictions

The following restrictions apply to the IMS Monitor Report Print utility:

- If the Monitor does not collect the types of information usually found in a particular report, that report, or the section of that report that would normally contain the information, is not produced. For example, if no checkpoints occur, only the headings for checkpoint are printed.
- In any report for which data is captured at the start and end of the Monitor trace interval, the report displays the data captured at these intervals, and their difference. Because data for these reports is needed at both intervals, these reports are not generated if the IMS control region is terminated prior to the termination of the Monitor trace.
- The Monitor must not be left on for more than 999 999 999 total DL/I calls if you plan to use Region Summary, Region Wait, Run-Profile, or Call-Summary (DB) reports. After 999 999 999 DL/I calls, truncation occurs in the various totals fields of these reports.

Most of the terms used in reports printed by the IMS Monitor Report Print utility (DFSUTR30) also appear in reports printed by the IMS Monitor Report Print utility (DFSUTR20).

Prerequisites

Currently, no prerequisites are documented for the DFSUTR20 utility.

Requirements

Currently, no requirements are documented for the DFSUTR20 utility.

Recommendations

Currently, no recommendations are documented for the DFSUTR20 utility.

Input and output

© Copyright IBM Corp. 1974, 2024

The Monitor Report Print utility runs as a batch program, with a sequential data set on DASD or tape as input. The contents of this data set are created by the IMS Monitor module (DFSMNTR0) in response to a **/TRACE SET ON MONITOR** command during IMS online execution.

JCL specifications

EXEC statement

Specifies the program name. The statement must be in the form:

```
// EXEC PGM=DFSUTR20,REGION=4096K
```

JOB statement

Initiates the job.

DD statements

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

//SYSPRINT DD
Specifies the output data set that is to contain the reports and control messages. It is usually coded as SYSOUT=A. The DCB parameters for this data set are RECFM=FBA and LRECL=133. BLKSIZE can be provided on the SYSPRINT DD statement and must be a multiple of 133. If the BLKSIZE is not provided, a default value of 133 will be used.

//SYSUT1 DD
Specifies the input data set to be analyzed. It is a labeled sequential data set written by the monitor module DFSMNTR0. (The ddname and dsname are IMSMON in the IMS procedure.)

//ANALYSIS DD
Specifies the Analysis Control data set. This file must be in card image format.

Analysis control data set

The Analysis Control data set determines which reports to print and allows for distribution redefinition for the Distribution reports.

- If you are printing only the Call Summary report, include the ONLY DLI statement in the Analysis Control data set for this run. The statement starts in card image column 1.
- To generate the Call Summary report, include the DLI statement in the Analysis Control data set for this run. If this statement is not included, the default option is taken; that is, all reports except the Call Summary report are printed. The statement starts in card image column 1.
- To generate the optional Distribution Appendix report, include the DIS statement anywhere in the Analysis Control data set. If this statement is not included, only the summary reports are printed. The statement starts in card image column 1.

If none of these options are selected, all reports except the Call Summary report and the Distribution Appendix report will be printed.

Specifying distribution redefinition

The general format for specifying a user redefinition of a distribution is:

```
Dn     n1,n2...
```

Dn
Starts in column 1 and is the distribution identifier (ID).

n1 through n9
Are each 8 digits or less, and each is a positive number between 0 and 99999999.

Each redefinition can occupy more than one statement, if necessary. The format for continuation statements follows the z/OS rules:

- The last value on the first statement must be followed by a comma and at least one blank.
- The first value on the continuation statement cannot start before column 2 nor after column 10.
- Comments can be included if they are preceded by at least one blank.

Assume that the distribution for region elapsed execution time is identified as D1 and has a default definition of:

```
0 1 2 3 30 300 3000 30000 3000000 30000000 INF
```

It can be redefined to be:

```
0 1 2 5 30 40 50 60 3000000 30000000 INF
```

This redefinition is accomplished by the following record in the Analysis Control data set:

```
D1 1,2,5,30,40,50,60,3000000,30000000
```

Because the numbers are positional parameters, the same redefinition could have been obtained by specifying the following:

```
D1 ,,5,,40,50,60
```

Related concepts

IMS Monitor reports (System Administration)

Related reference

Database-Monitor Report Print utility (DFSUTR30) (Database Utilities)

Examples of the DFSUTR20 utility

The following JCL produces a complete set of reports, including the Call Summary report, from a tape with a serial number of IMSDA1.

```
//TRACE    JOB
//*
//        EXEC PGM=DFSUTR20,REGION=512K
//STEPLIB   DD DSN=IMS.&SYS2..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 *
DLI CALL REPORT
DISTRIBUTION
/*
```

The following example shows how the distributions for D30 and D2 are modified if the JCL is modified.

```
       .
       .
       .
//ANALYSIS DD *
DLI CALL REPORT
DISTRIBUTION
D30 8000,24000,50000,75000
D2 1000,2000,3000,4000,5000,6000,7000,8000,9000
/*
```

Chapter 18. Log Transaction Analysis utility (DFSILTA0)

Use the Log Transaction Analysis utility (DFSILTA0) to collect information about individual occurrences of IMS transactions, based on records in the IMS log data set.

The information collected includes:

- Transaction identification
- Source
- Transaction program name
- Dependent region
- Priority
- Class of the transaction

Canceled messages are not used.

DFSILTA0 also accumulates:

- The time that each transaction is received
- The time of the message get unique (GU) call
- The time the transaction processing ends.
- The time the output message is placed on the output queue
- The time the output message starts to the terminal

From these times, DFSILTA0 calculates:

- Total response time
- Time on the input queue
- Processing time
- Time on the output queue

You can use this information to find bottlenecks in the system and to evaluate whether transaction classes have been assigned correctly. If you are running the Statistical Analysis utility on a smaller portion of the IMS log data, DFSILTA0 can provide a new log tailored to your specifications. DFSILTA0 is put into IMS.SDFSRESL during IMS system definition.

Subsections:

- "Restrictions" on page 459
- "Prerequisites" on page 460
- "Requirements" on page 460
- "Recommendations" on page 460
- "Input and output" on page 460
- "JCL specifications" on page 460

Restrictions

The Log Transaction Analysis utility has the following restrictions:

- Log data sets from a batch region are not used.
- Canceled messages are not used.

© Copyright IBM Corp. 1974, 2024

- DFSILTA0 creates a queue entry in a GETMAIN storage pool for each transaction that falls within the specified times or checkpoints. These queue entries are not freed nor are they reused until all the log records necessary to complete an entry on the log transaction analysis report are found on the log.
- If a large number of transactions are enqueued but not processed for any reason, an increase in storage usage and processor time can occur.
- Common Queue Server (CQS) logs cannot be used as input by the Log Transaction Analysis utility because CQS log records have a different format from IMS log records.
- The utility works only with input log data sets created by the same release of IMS as the utility release level.

Prerequisites

Currently, no prerequisites are documented for the DFSILTA0 utility.

Requirements

Currently, no requirements are documented for the DFSILTA0 utility.

Recommendations

Currently, no recommendations are documented for the DFSILTA0 utility.

Input and output

There are three types of input to DFSILTA0:

- IMS log data. This is required.
- Report title statement. This provides descriptive information for the optional title data set.
- Parameter. The optional input parameter ST=, which specifies what portion of the log data set is to be examined for transactions.

All of the DD statements to generate output for the DFSILTA0 utility are optional. If you do not specify any DD statements, the utility will waste CPU time and will not return any output. If you specify any of the DD statements, DFSILTA0 can produce the following output:

- A new data set containing the IMS log records for the time period processed
- A detailed report in input sequence, if the PRINTER DD statements is specified
- A report that can be sorted to produce a sequenced report (if the REPORT DD statement is specified)
- A heading report (if the HEADING DD statement is specified)

The starting position and length of the field names on the Detailed Report Format are used in the optional sort step to produce sequenced reports.

JCL specifications

The Log Transaction Analysis utility executes as a standard operating system job. You must define an EXEC statement and DD statements.

EXEC statement

Executes the Log Transaction Analysis utility, DFSILTA0.

```
//STEP0    EXEC PGM=DFSILTA0,PARM='ST'=(hhmmss+HHMM,,mm)
```

ST=

Specifies starting and ending times. If the ST parameter is omitted, the default is the first checkpoint encountered. The format of the ST= parameter is:

```
ST=ALL
   (hhmmss[{+|-}HHMM],
   c,mm,e)
```

Note that the ST= parameter has four positional parameters in addition to the ALL parameter. With the exception of the ALL parameter, these parameters must be enclosed in parenthesis.

ALL

Specifies the complete log data set.

hhmmss

Specifies an hour, minute, and second. Only transactions that originate after the first checkpoint occurring at or after this time are processed. The default is to process 10 minutes from this time.

Tip: This parameter is always assumed to refer to a time later than the first checkpoint on the input log. If you want to process transactions starting with the first checkpoint on the log, do not specify a value for this parameter.

{+|-}HHMM

Specifies the time-zone offset used to convert local time to Universal Time Coordinated (UTC) time.

+ or -

Specifies the sign of the offset. Can be blank only if hh and mm are also blank. The time zone is only needed if the offset to the UTC on the day entered is different from the current offset. One example would be if the offset was due to a daylight saving time change.

HH

Specifies hours of offset, a number from 0 to 14 or blank only if mm is also blank

MM

Specifies minutes of offset; can be 00, 15, 30, 45, or blank

If an offset of +|-0000 is specified, the starting time is UTC. If no offset is supplied, the offset is obtained from the z/OS offset.

C

Specifies the number of checkpoints to be processed before selection of transactions stops. C is a number from 1 to 9.

MM

Specifies the number of minutes to select transactions. MM is a number between 0 and 99.

E

Specifies to end of data set from the specified start time. E is the default.

The Log Transaction Analysis utility scans records between checkpoints. Records before the first checkpoint on an intermediate log data set would only be analyzed by reference to a checkpoint on a previous log.

DD statements

STEPLIB DD

Points to IMS.SDFSRESL, which contains the IMS nucleus and required utility modules.

```
//STEPLIB DD DSNAME=IMS.SDFSRESL,DISP=SHR
```

HEADING DD

Describes the optional heading output data set.

```
//HEADING  DD  SYSOUT=A
```

PRINTER DD
Describes the optional printed report output data set.

```
//PRINTER  DD  SYSOUT=A
```

REPORT DD
Describes the optional report output data set. This data set can pass to a sort step. Report entry headings and any checkpoint records are not included in this data set.

```
//REPORT  DD  DSN=&&REPORT,DISP=(,PASS),UNIT=SYSDA,
//            SPACE=(CYL,(1,1))
```

LOGINxxx DD
Describes the input log data sets. The last three characters are optional. If more than five characters are used, the last character will be used in the report as the system ID.

Log data from each IMS system must be allocated to a single DD statement. If multiple log data sets from an IMS system are used, they must be contiguous and they must be concatenated in the order in which they were created.

```
//LOGIN  DD  DSN=IMS.LOG,DISP=OLD,VOL=SER=XXXXXX,
//           UNIT=YYYY
```

LOGOUT DD
Describes the optional log data set. This log data set can be used as input to the Statistical Analysis utility.

The LOGOUT data set content is identical to that of LOGIN within the interval specified, except that the type 6 record at the beginning of LOGIN is recopied.

```
//LOGOUT  DD  DSN=IMS.LOGOUT,DISP=(,PASS),
//            VOL=SER=XXXXXX,UNIT=TAPE,DCB=(RECFM=VB,
//            LRECL=6004,BLKSIZE=6008)
```

TITLE DD
Describes the optional title data set. This allows for the inclusion of descriptive information on each page of the printer output data set.

```
//TITLE  DD  *
  * * * Descriptive information
```

The SORT step is optional. It is used to produce sequenced reports.

EXEC
Executes the sort program.

```
//STEP1  EXEC  PGM=SORT
```

SYSOUT DD
Describes the message output data set for the sort.

```
//SYSOUT DD SYSOUT=A
```

SORTIN DD
Describes the input data set to the sort. It is the data set described by the REPORT DD statement.

```
//SORTIN  DD  DSN=&&REPORT,DISP=(OLD,DELETE)
```

SORTOUT DD
Describes the output data set to the sort. It is used for printing a sequenced report.

```
//SORTOUT  DD SYSOUT=A
```

SORTWK01-12|DD

Describe the sort program's work data sets. At least three data sets must be used. They can be tape or disk. For disk the format is:

```
//SORTWKnn  DD  UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
```

SYSIN DD

Describes the sort program's control data set. For a control data set in the input stream, the format is:

```
//SYSIN  DD  *
```

Example The following is a sample SORT control statement that provides a report sequenced by message get unique (GU) schedule time within a region:

```
  SORT FIELDS=(67,7,CH,A,55,2,CH,A),SIZE=E500
```

Related concepts

Statistical-analysis, log-transaction reports, and analyzing log records (System Administration)

Related reference

“Statistical Analysis utility (DFSISTS0)” on page 471

Use the Statistical Analysis utility (DFSISTS0) to analyze the information in any of the IMS system logs, except those from a batch region.

Chapter 19. Offline Dump Formatter utility (DFSOFMD0)

Use the Offline Dump Formatter utility (DFSOFMD0) to format internal IMS control blocks in a dump that is both independent of a failure and independent of the dumping process.

This utility allows you to tailor the dump to print and format only the data areas needed to analyze a particular problem. Use the Offline Dump Formatter utility to:

- Establish the environment needed for offline dump formatting
- Read and check the dump format control statements
- Relocate or load the dump formatting modules
- Direct the offline dump formatting process

The Offline Dump Formatter utility is invoked as a verb exit from the Interactive Problem Control System (IPCS).

The Offline Dump Formatter utility modules are included in the dumped storage to ensure that the modules used for formatting the dump match the level of the dumped IMS control blocks. These modules can be relocated from the dumped storage, or a fresh copy can be loaded from the program library.

The Offline Dump Formatter utility can be used even if you have more than one release level of IMS, or if you are using any supported version of IMS. The load modules for the Offline Dump Formatter utility are associated with aliases that allow IMS.SDFSRESL from different releases to be concatenated in IPCS TASKLIB. The aliases are:

Alias
Load Module

DFSOF111
DFSOFMD0

DFSAB111
DFSABND0

The IPCS TASKLIB concatenation can contain multiple execution libraries.

Subsections:

- “Restrictions” on page 465
- “Prerequisites” on page 465
- “Requirements” on page 466
- “Recommendations” on page 466
- “Input and output” on page 466

Restrictions

The following restrictions apply to the Offline Dump Formatter utility:

- You cannot use the Offline Dump Formatter utility for batch regions that are not currently producing IMS online formatted dumps, such as the Pre-reorganization utility and the Image Copy utility, because they do not contain the required IMS control blocks for IMS dump formatting.

Prerequisites

Currently, no prerequisites are documented for the DFSOFMD0 utility.

© Copyright IBM Corp. 1974, 2024

Requirements

The following restrictions apply to the Offline Dump Formatter utility:

- The machine that executes this utility must be licensed to run IMS.
- The Offline Dump Formatter utility is conditionally assembled during IMS control block generation because of dependencies on z/OS services for GETMAIN, ESTAE, and LOAD. If the DFSOFMD0 module is loaded with LOAD SVC by IPCS, the module must be in the STEPLIB data set or in linklist libraries.
- The DFSOFMD0 module must be at the same release level as the IMS system it is formatting. It must be assembled on a z/OS that is the same level as the z/OS it is formatting. This condition applies even if you concatenate an IMS.SDFSRESL from a previous release.
- The version of IPCS you use to execute this utility must be compatible with the z/OS system that was dumped.
- SYS1.DUMPxx data sets must be large enough to contain a complete dump of the IMS control region, DL/I, DBRC, and IRLM address spaces for systems using the IMS SDUMP option.
- To format Fast Path Dumps, you need to use formatting modules from an IMS system generated with Fast Path.
- If you are using IMS Shared Message Queues or Shared EMH Queues, then your SYS1.DUMPxx data sets must be large enough to contain a dump of the CQS address space in addition to the address spaces. If you are using the Common Service Layer (CSL), then your SYS1.DUMPxx data sets must be large enough to contain a dump of the SCI address space in addition to the address spaces.

Recommendations

Currently, no recommendations are documented for the DFSOFMD0 utility.

Input and output

This utility requires the following input:

- An acceptable machine-readable dump, such as:
  - SDUMP
  - SYSMDUMP
  - Stand-alone dump
  - Dump requested by the z/OS DUMP command
  - Any other machine readable dump of the IMS system address spaces

  The dump must include key 0 and key 7 CSA, the CVT, SQA, and at least one of the CTL or DL/I SAS address spaces. CSA is not required in a batch environment.
- An IMS dump format control data set or FMTIMS (options) specified on the IPCS VERBX control statement.
- Execution of a proper VERBX control statement for IPCS.

The output for this utility is a formatted dump of specified sections of an IMS dump.

If you are using Offline Dump Formatter with an execution library that is from an earlier IMS release, a formatter dialog initialization warning occurs if CSA is not included with batch SYSMDUMPs. Offline Dump Formatter cannot determine the release levels for the concatenated program libraries, but continues under the presumption that they are correctly concatenated.

Running the DFSOFMD0 utility

Using the IMS Dump Formatter gives you a menu-driven way to run the Offline Dump Formatter utility without complicated editing of the DFSFRMAT file.

IPCS uses menus on the screen to run the IMS Dump Formatter. These menus allow you to specify the information to be contained in the dump. The IMS Dump Formatter calls the Offline Dump Formatter utility to perform the required formatting tasks. The output is returned in a format that you can read on the terminal.

Running the utility in an IMS online environment

To use the Offline Dump Formatter utility in IMS DB/DC, DCCTL, or DBCTL environments, specify the IMS start parameter option FMTO=D.

You can also use a SYSMDUMP DD statement.

Running the utility in an IMS batch environment

To format IMS batch job dumps offline in DBCTL, DB/DC, or DCCTL batch environments, you can request a z/OS SYSMDUMP. z/OS creates a dump that can be formatted offline using the IMS Offline Dump Formatter utility. Before using the utility, you must remove the SYSUDUMP or SYSABEND DD statement in the IMS batch JCL procedures and insert a SYSMDUMP DD statement.

If the SYSMDUMP data set is too small, unavailable, or unusable, the operating system might be unable to make a usable dump of the batch job.

IPCS execution

To use the Offline Dump Formatter utility under IPCS, you must provide an IMS user control statement.

Some examples of the IMS user control statement include:

```
VERBX DFSOFMD0 'jjjjjjjj[,R][,D]' verbx_options
VERBX DFSOFMD0 'jjjjjjjj[,R][,H],FMTIMS(ALL)' verbx_options
VERBX DFSOF320 'jjjjjjjj,FMTIMS(SCD)' verbx_options
VERBX DFSOF320 'jjjjjjjj[,R][,N],FMTIMS(AUTO,MIN)' verbx_options
VERBX IMSDUMP 'jjjjjjjj[,R][,D],FMTIMS(SAVEAREA,DISP)' verbx_options
VERBX IMSDUMP 'jjjjjjjj[,R][,D]' verbx_options
VERBX IMSDUMP 'IMSDUMMY,R,FMTIMS(LOG)' verbx_options
VERBX IMSDUMP 'IMSDUMMY,R,FMTIMS(SAP,2A723C80)' verbx_options
VERBX IMSDUMP 'IMSDUMMY,R,FIMTIMS(SAVEAREA,SUM)' verbx_options
```

The control statement parameters are:

jjjjjjjj
Indicates the job name or started task name of either the IMS CTL, DL/I, or the IMS batch address space.

R
Indicates REFRESH, an optional parameter for requesting that the IMS dump formatter modules be loaded from current program libraries. If you do not specify R, and invalid dumped formatter routines still exist, the invalid routines might be loaded instead of the current libraries.

H
Indicates HALFLINE, an optional parameter to request that the IMS dump formatter be limited to the width of a screen (that is, 80 characters per line).

N
Indicates NO HEADER, an optional parameter that reduces the header print volume when formatting small data area dumps. The formatter skips the printed header and footer and suppresses the dump content warning messages that describe missing IMS address spaces or address spaces that did not finish initializing.

D
Indicates DEBUG, an optional parameter for requesting that the IMS offline formatter not create its ESTAE and thereby allow a dump of any IMS dump formatter abend.

FMTIMS(options)
Specifies the FMTIMS verb. The FMTIMS verb must be specified in either the control statement or in the IMS dump format control data set description (DFSFRMAT DD). FMTIMS permits a subset of formatting options that describe the sections of the IMS dump to be formatted during the current pass of IPCS. The DFSFRMAT DD description describes this subset.

verbx_options
Are valid IPCS VERBX command options.

If you do not specify FMTIMS in the user control statement, you must provide an IMS dump format control statement with DFSFRMAT options specified.

The following example is of a TSO ALLOCATE command to provide IMS dump format control data set information:

```
ALLOC FI(DFSFRMAT) SHR DA('dump.control.dsname')
```

DD statements

INDEX DD
Allows the dump index to print ahead of the formatted dump.

DFSFRMAT DD
Describes an IMS dump format control data set. The data set contains control statements that specify the sections of the IMS dump to be formatted during the current pass of IPCS. If this statement is not specified, the formatting option defaults to SUMMARY.

The IMS dump format control data set is a sequential data set that must be defined with a fixed or fixed-blocked record format (RECFM=F or FB). The record length can be any valid size. The data set contains an FMTIMS verb, followed by subset options describing the sections of IMS to be formatted. You can request a short version of the formatted subset by adding the MIN parameter to the option you select.

You can allow IMS to select the dump formatter options for you by specifying the AUTO option. When you specify AUTO, IMS determines the options to use by looking at the ITASKs that are failing and by selecting the appropriate sets of options for the required dump formatter output. You can specify AUTO with MIN or SUM qualifiers. If you use MIN or SUM, the qualifier is added to each option that AUTO selects.

Subset options can be specified in any combination and in any order. The following subset options can be specified independently or can be qualified as shown, but require no additional arguments:

- ALL or (ALL,MIN)
- AOI Automated Operator Interface (Directed Message Manager)
- AUTO, or (AUTO,MIN), or (AUTO,SUM)
- CBT
- DB or (DB,MIN)
- DBRC

- DBRM Database Recovery Manager
- DC or (DC,MIN)
- DEDB or (DEDB,MIN)
- DISPATCH or (DISPATCH,MIN)
- EMH or (EMH,MIN)
- IRLM control block formatting
- LOG or (LOG,MIN)
- LUM
- MSDB or (MSDB,MIN)
- OTMA Open TM Access
- QM or (QM,MIN)
- RESTART
- SAVEAREA, or (SAVEAREA,MIN) or (SAVEAREA,SUM)
- SB or (SB,MIN)
- SCD or (SCD,MIN)
- SDE Storage Descriptor Element Blocks and Storage
- SMBS All SMBs
- SPST
- SUBS
- SUMMARY or (SUMMARY,MIN)
- SYSTEM or (SYSTEM,MIN)
- UTIL

The following subset options require additional arguments or qualifications as shown:

- (CBTE,cbteid)
- (CLB,address) or (CLB,nodename) or (CLB,lterm name) or (CLB,comm id)
- (DPST,address) or (DPST,number) or (DPST,name)
- (LLB,link number)
- (LUB,lu name)
- (POOL,poolid) or (POOL,poolid,MIN)
- (SAP,sapaddr) or (SAP,ecbaddr)
- (SYSPST,system pst address) or (SYSPST,system pst name)
- (TRACE,name) or (TRACE,name,MIN)

Related concepts

z/OS: MVS interactive problem control system (IPCS) introduction

Solving IMS problems by using the IMS Offline Dump Formatter (Diagnosis)

Chapter 20. Statistical Analysis utility (DFSISTS0)

Use the Statistical Analysis utility (DFSISTS0) to analyze the information in any of the IMS system logs, except those from a batch region.

The DFSISTS0 utility resides in IMS.SDFSRESL data set.

To run the Statistical Analysis utility on a selected portion of an IMS system log, a new log that is tailored to your own specifications can be created by using the Log Transaction Analysis utility.

Subsections:

- "Restrictions" on page 471
- "Prerequisites" on page 471
- "Requirements" on page 471
- "Recommendations" on page 471
- "Input and output" on page 471
- "JCL specifications" on page 475
- "Utility control statements" on page 477

Restrictions

The Statistical Analysis has the following restrictions:

- Common Queue Server (CQS) logs cannot be used as input because CQS log records have a different format from IMS log records.
- The utility works only with input log data sets created by the same release of IMS as the utility release level.

Prerequisites

Currently, no prerequisites are documented for the DFSISTS0 utility.

Requirements

Currently, no requirements are documented for the DFSISTS0 utility.

Recommendations

Currently, no recommendations are documented for the DFSISTS0 utility.

Input and output

Log records

The following log records are used by the IMS Statistical Analysis utility:

01
Input message ready to be put on the destination message queue

© Copyright IBM Corp. 1974, 2024

03
Output message segment ready to be put on the destination message queue

07
An application program has been terminated

31
The application program issues a "get unique" to retrieve its next message

34
Message canceled and a portion of that message has been previously logged

35
Message has been put on the destination message queue

36
Message has been taken off the destination message queue

48
A variable-length padding log record

The following list provides a detailed explanation of each log record type.

Log Type 01
Log Type 01 record is written when a message is completely received by communications and is ready to be put on the destination queue. The destination queue is either a Scheduler Message Block (SMB) or Communications Name Table (CNT). The SMB destination means a transaction code has been entered by the terminal operator, and an application program will be scheduled. A CNT means a message switch will be done. If the terminal operator entered an LTERM and a message, no application program is necessary. The message will be queued for output directly on the LTERM named in the input message.

Log Type 03
When a segment of a message has been created by an application program and is ready to be put on the destination message queue, the 03 record is written. The destination message queue can be either on SMB or CNT. If SMB is the destination, a "program-to-program" message switch is called for by the application program. If the segment is destined for a CNT, the application program is sending an output message to an LTERM.

In a type 03 record, the date and time fields, PDATE and PTIME, are carried forward from the 01 record. When the statistics utilities are run, the 03 records and 36 records are correlated to determine response time. The time reflected is from the time the message is put on the input queue (obtained from the 03 record) until the message is released from the output queue (obtained from the associated 36 record).

Log Type 07
This record is the application accounting record of the system. The type 07 record is written when an application program terminates in a message processing or batch-message processing region.

Log Type 31
The 31 record is written when the application program issues a "get unique" to retrieve its next message.

Log Type 34
A type 34 record is written when a message has been canceled and a portion of that message has already been logged.

Log Type 35
The 35 record is written when a message (input or output) has been put on the destination queue.

If the message is very long and requires more than one input message buffer, the record has the date and time in it. The date and time in the type 01 record is invalid under this condition.

Log Type 36
A type 36 record is written when a message has been sent in its entirety and the message is ready to be released from the queue. On all devices except display devices, the message is ready to be released from the queue as soon as the last segment is successfully sent to the terminal. Display

devices are different. If the display output is only a single page, the message is dequeued after the last segment has been successfully sent.

For multiple pages of display output, the PAGEDEL option selected on the TERMINAL macro at system definition time determines when the message is ready to be released from the queue. If you specify option=PAGEDEL (or PAGEDEL=YES), the message is dequeued when you enter a question mark, PA2 key, or a new input transaction. Option=NPGDEL (or PAGEDEL=NO) requires you to enter a question mark or PA2 key to take the message off the output queue and write the type 36 record.

The effect of option=NPGDEL (or PAGEDEL=NO) on response time can be dramatic. If you leave the current message displayed for a long period or power off the video device, the message is not removed from the output queue and the type 36 record is not written until terminal operations begin again. Consequently, response time appears to require many hours or even days.

Log Type 48

A type 48 record is a variable-length padding log record that contains the time zone offset from GMT time.

Reports

Reports are generated if the PRINTDCB DD statement is included. The different types of statistical reports are described as follows:

- Messages Queued but Not Sent—by destination

  The output message (X'03') appears on the log, but no record (X'36') appears to indicate that the message was sent to the terminal. Output is sorted by symbolic terminal name.

- Messages—Program to Program—by destination

  An output message (X'03') is sent to an SMB. Output is sorted by destination.

- Line-and-Terminal Report

  The line-and-terminal report shows the line and terminal loading by time of day (which can be used to determine the line and terminal utilization, peak traffic periods, and so forth).

  The line-and-terminal report counts input messages (R), X'01', to IMS from each LTERM, and output messages (S), X'03', to each LTERM from IMS.

  – A message switch counts as two messages; one from the originating terminal, one to the destination terminal.
  – A broadcast message counts as one message from the originating terminal, and one message each to the destination terminals.

The next four reports deal with transaction codes. If an output message is generated by a command from a different terminal, the input prefix data is replaced by the message "THIS OUTPUT NOT RESULT OF INPUT." An X'03' message generated by the system, independent of terminal input, has a transaction code of IMSSYS. If an output message was generated by an input message that was not on the log or by a command from the same terminal (for example, DISPLAY), the transaction code is NOTAVA; otherwise, the transaction code can be found in the generating X'01' log record.

- Messages Queued but Not Sent—by transaction code

  The output message (X'03') appears on the log, but no record (X'36') appears to indicate that the message was sent to the terminal. Output is sorted by transaction code.

- Messages—Program to Program—by transaction code

  An output message (X'03') was sent to an SMB. Output is sorted by transaction code.

- Transaction Report

  This report shows loading by transaction code and by time of day.

  – An "R" indicates the time of day an input message was inputted to IMS from each logical terminal.
  – An "S" indicates the time of day an output message was outputted to each logical terminal.

The report counts the same messages as the Line-and-Terminal Report. Input is sorted by transaction code.

The transaction code column can contain the following entries:

(NOSORC)
The output message was generated by a command.

(NOTAVA)
The output message was generated by an input message that was not on the input log.

(IMSSYS)
The output message was generated by IMS.

- Transaction-Response Report

  Measures two response times. The first line is response time from complete receipt of the input message (enqueue time X'35') until the response message to the terminal is successfully dequeued (X'36'). The second line is response time from complete receipt of the input message (enqueue time X'35') until the response message to the terminal is started (GU time X'31').

  There can be multiple responses from a single transaction, and they can include any output messages from program-to-program switch transactions that are a result of the original input message.

  The percentile report shows shortest response, longest response, and 25th, 50th, 75th, and 95th percentile response. A response time within the n^{th} percentile is greater than or equal to n% of the total number of response times processed for that transaction code. For example, a 04.3S number under the '75% RESPONSE' column means that 75% of the total responses for that transaction were equal to or less than 04.3 seconds.

- Application-Accounting Report

  Provides sufficient data to allow machine charges to be distributed among application programs or transaction codes.

  The following information is contained in this report:

  – Counts of all requests to DL/I
  – Amount of processor task time

  All requests for services from DL/I, for access to messages or databases, are counted. These counts are accumulated by program, by transaction code within program, and by priority within transaction code.

  Counts of messages processed and of "get uniques" issued are included. The counts will be different because "get unique" is a request for messages that may return several files and a non-zero return code. The messages processed would not include the return code.

  CPU time is set when a request for scheduling is made. The value is the maximum time for each transaction, multiplied by the maximum number of transactions. The remaining time is requested prior to the next request for scheduling. This time is the actual time the program executed. It does not include any wait time for accessing data. This time can be incorrect if the application program is a BMP and issues a TTIMER or STIMER macro.

  Average processor time is the total message processor time, divided by the number of messages. It is not rounded. The final average processor time is a recalculated average.

  Number of bad completion codes reflects the number of times an application program terminates abnormally or returns with a value other than zero in register 15.

- IMS-Accounting Report

  Shows start and stop times for the IMS control region.

Message Select and Copy or List option

The execution of the Message Select and Copy or List is optional.

The Message Select and Copy or List program is executed if either of the IMSLOGO or IMSLOGP DD statements is included.

Messages can be listed or copied in order of transaction, terminal, or time entered.

JCL specifications

EXEC statement

Executes the Statistical Analysis utility, DFSISTS0.

```
//STEP1    EXEC PGM=DFSISTS0,PARM='LINECNT=XX'
```

If LINECNT is not specified, the default is 36.

DD statements

STEPLIB DD

Describes the program library containing the utility programs. Its format is:

```
//STEPLIB DD DSNAME=IMS.SDFSRESL,DISP=SHR
```

//LOGINxxx DD

Describes the input log data sets. The last three characters are optional.

Log data from each IMS system must be allocated to a single DD statement. If multiple log data sets from an IMS system are used, they must be contiguous and they must be concatenated in the order in which they were created. The format of the LOGINxxx DD statement is:

```
//LOGIN DD DSNAME=IMSLOG,DISP=OLD,VOL=SER=XXXXXX,UNIT=TAPE
```

SORTWK01-32 DD

Describes the sort program's work data sets. The space defined can vary. The number of data sets must be at least three. They can be on either tape or disk. For a disk sort, the format is:

```
//SORTWKnn DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
```

SORTLIB DD

Describes the library containing the sort program's modules. Its format is:

```
//SORTLIB DD DSNAME=SYS1.SORTLIB,DISP=SHR
```

SORTOUT DD

Does not use the output data set. However, the DCB information must appear on the DD statement. Its format is:

```
//SORTOUT DD DUMMY,DCB=*.LOGIN
```

SYSPRINT DD

Describes the output data set for control messages. Its format is:

```
//SYSPRINT DD SYSOUT=A
```

The next step is a sort, and all DD statements, with the exception of SORTIN and SORTOUT, are the same as the previous sort.

PRINTDCB DD

Describes the report output, normally the output stream. It can be blocked or unblocked, because I/O is performed using QSAM, with QSAM acquiring the buffers. This DD statement is optional. The format of the PRINTDCB DD statement is:

```
//PRINTDCB DD SYSOUT=A,DCB=(BLKSIZE=133,
//            LRECL=133,RECFM=FA)
```

IMSLOGP DD

Describes the message select and list output. This DD statement is optional. The format of the statement is the same as the PRINTDCB statement for DFSIST30.

IMSLOGO DD

Describes the message select and copy output. This DD statement is optional. The format of the IMSLOGO DD statement is:

```
// IMSLOGO DD DSNAME=OUTPUT,UNIT=tttt,
//            DISP=(NEW,KEEP),DCB=(RECFM=VB,
//            LRECL=32756,BLKSIZE=32760)
```

The IMSLOGO DD statement specifies an output data set that contains the text segments from the message records, type 01 and type 03 log records, used by the DFSISTS0 utility. The following table shows the broken down sort fields that are contained in the first 93 bytes (X'5D') of each record.

Table 26. First 93 byte (X'5D') sort fields of the IMSLOGO DD statement

Offset (Hex)	Length	Description
0	4	Length of the IMSLOGO record
4	4	Gregorian date (YYYYMMDD)
8	1	The constant X'04'
9	9	Zeros
12	8	Time the input message was created
1A	8	Time the output message was created
22	2	Internal sequencing field
24	1	Type of message: **01** A message to IMS from an external source, such as a terminal **02** A program switch, where one transaction invokes another transaction **03** A message destined for an external target, such as a terminal
25	8	Zeros
2D	8	Transaction code
35	8	Input terminal ID
3D	4	Sequence number of this message from the input terminal ID
41	8	Message input time

Table 26. First 93 byte (X'5D') sort fields of the IMSLOGO DD statement (continued)

Offset (Hex)	Length	Description
49	8	Output terminal ID
51	4	Sequence number of this message to the output terminal ID
55	8	Message output time

Starting at offset X'5D' is the message text, starting with the 2-byte length. The message text is truncated as necessary to fit into a record that cannot exceed the LRECL of the IMSLOGO data set, 32,756 bytes (X'7FF4').

Utility control statements

The Message Select and Copy or List program selects messages based on control statements read from SYSIN. Messages selected are printed or copied onto an output data set. If the DD statement IMSLOGO is included, an output data set is created. If the DD statement IMSLOGP is included, messages selected are printed. Both DD statements can be included in a single run.

The following restrictions apply to the control statements:

- All control statements begin in position 1, with a keyword identifying that control statement.
- Following the keyword is a series of parameters, enclosed within parentheses and separated by commas.
- Control statements cannot be continued beyond position 71.
- Multiple control statements with the same keyword, starting in position 1, are permitted.
- Within parentheses, all parameters are positional; missing parameters must be indicated by commas.
- Messages are selected if they fulfill at least one of the criteria specified by the control statement.

A group of names can be indicated by terminating the parameter with an *.

Example: INV* causes the names INV, INVENTORY, INVA, and INVB to be selected.

The name parameter ALL can be used to select all names rather than a specified name.

Transaction code control statement

The format of the transaction code control statement is:

```
TRANS CODE=(TRANSCOD,I,O),(TRANSA,I),(INV*,,O),(ALL,I,O)
```

- The first parameter is a transaction code from 1 to 8 characters.
- The second parameter, I, selects input messages with this code.
- The third parameter, O, selects output messages resulting from this code.
- A transaction code of ALL selects all transaction codes.
- An asterisk within the transaction code causes only characters preceding the asterisk to be compared with the corresponding number of characters from the input transaction code to determine selection. You can use this to select groups of transaction codes.

Symbolic terminal name control statement

Examples of the symbolic terminal name control statement are:

```
SYM NAME=(TERMA,I,O,ALL),(TERM*,I,,ALL),(TERMINV,,O,ALL)
SYM NAME=(TERMPAY,I,O,TERMA)
SYM NAME=(ALL,,O,TERMA)
```

- The first parameter is a symbolic terminal name of a source of input messages that can be from 1 to 8 characters in length.
- The second parameter, I, selects input from this terminal.
- The third parameter, O, selects output generated by input from this terminal.
- The fourth parameter is a symbolic terminal name of a destination of output messages that can be from 1 to 8 characters in length.
- A symbolic terminal name of ALL selects all symbolic terminal names
- An asterisk within the terminal name causes only characters preceding the asterisk to be compared with the corresponding number of characters from the input terminal name to determine selection. You can use this to select groups of terminal names.

Time control statement

The format of the time control statement is:

```
TIME=(yyddd,hhmm[{+|-}HHMM],yyddd,hhmm[{+|-}HHMM])
```

- The first parameter is starting date—year (YY) and day of year (DDD).
- The second parameter is starting time—hours (HH) and minutes (MM) plus the optional time-zone offset information—{+|-}HHMM.
- The third parameter is ending date—year (YY) and day of year (DDD).
- The fourth parameter is ending time—hours (HH) and minutes (MM) plus the optional time-zone offset information—{+|-}HHMM.

  The optional time-zone parameters used in the second and fourth parameters are as follows:

  – The {+|-} is the sign of the time-zone offset from Universal Time Coordinated (UTC).
  – The HH is the number of whole hours of offset from UTC.
  – The MM is the minutes of offset; can be 00, 15, 30, 45, or blank.
- If you include the time control statement, only messages specified by a transaction code statement or a terminal control statement and falling within the specified times are selected.

Nonprintable character control statement

The format of the nonprintable character control statement is:

```
NON PRINT=HEX
```

If you include this control statement, nonprintable characters are printed in hexadecimal, on two lines, with one hexadecimal character above the other. If you do not include this statement, nonprintable characters appear as blanks.

Message Select Output Order Statement

The format of the message select output order statement is:

```
ORDER=CREATE|SOURCE|DEST
```

When messages are listed, the default order is to list in order of the time of the first input message ("first" means first of a group of messages that result from that message. This group usually consists of an input

message and its response, but intermediate program switches are included if they exist). This default order is CREATE.

To list messages in order by LTERM of the first input message, one specifies SOURCE and to list messages in order by the target of the initial message (the initial transaction for anything but a message switch), one specifies DEST.

Related concepts

Statistical-analysis, log-transaction reports, and analyzing log records (System Administration)

Related reference

"Log Transaction Analysis utility (DFSILTA0)" on page 459

Use the Log Transaction Analysis utility (DFSILTA0) to collect information about individual occurrences of IMS transactions, based on records in the IMS log data set.

Examples of the DFSISTS0 utility

The following examples show the output that is generated by running the Statistical Analysis utility (DFSISTS0) with either the //PRINTDCB DD statement, the //IMSLOGP DD statement, or both.

JCL for the statistical analysis utility

The following figure is an example of the JCL for execution of the Statistical Analysis utility. This is a full statistics, job-stream example with sorting by date that produces reports under date control. BLKSIZE and LRECL in all data sets are dependent on the input log.

```
//STATS    JOB 1,NAME,MSGCLASS=A,MSGLEVEL=1,PRTY=8
//STEPLIB   DD DSNAME=IMS.SDFSRESL,DISP=SHR
//*
//ST1     EXEC PGM=DFSISTS0
//SORTLIB   DD DSN=SYS1.SORTLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//SYSUDUMP  DD SYSOUT=A
//LOGIN     DD DSN=IMSLOG,DISP=OLD,
//             UNIT=TAPE,VOL=SER=LOGTAP
//SORTWK01  DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTWK02  DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTWK03  DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTOUT   DD DUMMY
//PRINTDCB  DD SYSOUT=A
//IMSLOGP   DD SYSOUT=A
//SYSIN     DD *
TRANS CODE=(ALL,I,O)
NON PRINT=HEX
/*
//
```

Statistics reports examples

The following list includes the statistics reports produced by DFSISTS0. Examples of the reports follow. The report date, which appears in the upper right corner of the following examples, will not appear unless a sort by date is specified as it was in the examples. The reports produced are:

- Messages—Queued but Not Sent by Destination ("Messages—Queued But Not Sent (by destination)" on page 480)
- Messages—Program to Program by Destination ("Messages—Program To Program (by destination)" on page 480)
- Line and Terminal ("Line-and-Terminal Report" on page 481)
- Messages—Queued but Not Sent by Transaction Code ("Messages—Queued But Not Sent (by transaction code)" on page 482)
- Messages—Program to Program by Transaction Code ("Messages—Program To Program (by transaction code)" on page 483)

- Transaction (“Transaction Report” on page 483)
- Transaction Response (“Transaction-Response Report” on page 485)
- Application Accounting (“Application Accounting Report” on page 486)

Using the normalized scientific notation in examples

When the number in the statistics report is too large to fit in the designated space, IMS uses the normalized scientific notation in place of the entire number.

In the normalized scientific notation, the character E is used to mean "times 10 raised to the power of *X*".

For example, 25,000 could also be written as 2.5E4 and 3.7E5 is equal to 370,000.

Messages—Queued But Not Sent (by destination)

```
     M E S S A G E S -- Q U E U E D   B U T   N O T   S E N T      D A T E
05/15/09           P A G E 00001
                    TOTAL
 DESTINATION      MESSAGES
 AZC0GH00                1
 CWARENS1                1
 DFSMTCNT                2
 F@MYDV21                1
 OQCCM211                1
```

Messages—Program To Program (by destination)

```
     M E S S A G E S -- P R O G R A M   T O   P R O G R A M        D A T E
05/15/09           P A G E 00001
                    TOTAL
 DESTINATION      MESSAGES
 VGG034T1                2
 VJDBBAT                32
 VJDBI4T                 3
 VJDBL1T                72
 VJDBL3T                 4
 VJDBL4T                39
 VJDBMLR                11
 VJDBML2                14
 VJDBR1T                74
 VJMPCC1                 1
 VJMPCC1B                8
 VJMPCC1C                4
 VJMPCC1D                7
 VJMPCC1E                5
 VJMPCC1F                3
 VJMPCC2               118
 VJMPCC3                 8
 VODSWC                  1
 VODSWT                 73
 VOGSF07                 4
 VOMPCC1                 5
 VOMPCC1A               13
 VOMPCC1B                9
 VOMPCC1C               14
 VOMPCC1D                3
 VOMPCC1E                1
 VOMPCC1F                2
 VOMPCC2C              174
 VOMPCC3                 2
 VOSSADD                72
 VOSSBIR                 2
 VOSSFDI                17
```

```
VOSSIHR                  2
VOSSLOG                  2
VOSSLPT                  1
VOSSNFH                 38
VOSSNMA                 31
VOSSSCR                  9
VOSSSMI                  4
VOSSSUP2                 1
VOSSTDC                  1
VOS027T1                 2
VOTABAD                  8
VOTAFDI                  3
VOTAFDO2                 8
VOTAMDS                 12
VOTANSA                  1
VOTAPAS                  2
VOTAPTA                  2
VOTARFT                  1
VOTASEB                 10
VOXPECIN                 1
YXCAJFX                  1
```

Line-and-Terminal Report

```
     L I N E   A N D   T E R M I N A L   R E P O R T              D A T E  05/15/09          P A G E 00001
                TOTAL        TOTAL          AVG                  HOURLY       DISTRIBUTION
 NODE    R/S   MESSAGES    CHARACTERS      SIZE  00-07 07-08 08-09 09-10 10-11 11-12 12-13 13-14 14-15 15-16 16-17
17-18 18-19 19-24
AADUGNA4 R             5         8,675     1,735     0     0     0     0     0     0     2     3     0     0     0
0     0     0
         S             5         8,675     1,735     0     0     0     0     0     0     2     3     0     0     0
0     0     0
ACRADDO1 S             3        25,194     8,398     0     0     0     0     0     0     3     0     0     0     0
0     0     0
ADEHORN0 R             1         1,735     1,735     0     0     0     0     0     0     0     1     0     0     0
0     0     0
         S             2         8,675     4,337     0     0     0     0     0     0     0     2     0     0     0
0     0     0
ADEHORN1 R             1         3,807     3,807     0     0     0     0     0     0     0     1     0     0     0
0     0     0
         S             2         3,807     1,903     0     0     0     0     0     0     0     2     0     0     0
0     0     0
ADHOWAR0 R             2         5,718     2,859     0     0     0     0     0     0     1     1     0     0     0
0     0     0
         S             2         5,718     2,859     0     0     0     0     0     0     1     1     0     0     0
0     0     0
ADHOWAR3 R             1         1,886     1,886     0     0     0     0     0     0     1     0     0     0     0
0     0     0
         S             1        11,316    11,316     0     0     0     0     0     0     1     0     0     0     0
0     0     0
ADHOWAR4 R             2         6,908     3,454     0     0     0     0     0     0     1     1     0     0     0
0     0     0
         S             2         6,908     3,454     0     0     0     0     0     0     1     1     0     0     0
0     0     0
ADOVALI0 R             2         3,454     1,727     0     0     0     0     0     0     2     0     0     0     0
0     0     0
ADUNCAN0 R             1         1,800     1,800     0     0     0     0     0     0     1     0     0     0     0
0     0     0
         S             1         3,600     3,600     0     0     0     0     0     0     1     0     0     0     0
0     0     0
AJWHIT10 R             3         2,233       744     0     0     0     0     0     0     3     0     0     0     0
0     0     0
         S             2           253       126     0     0     0     0     0     0     2     0     0     0     0
0     0     0
AKCAMPB0 S             8        13,880     1,735     0     0     0     0     0     0     4     4     0     0     0
0     0     0
ALOPEZ2  R             4         5,852     1,463     0     0     0     0     0     0     2     2     0     0     0
0     0     0
ALUOMA1  R             1         1,758     1,758     0     0     0     0     0     0     1     0     0     0     0
0     0     0
         S             1         2,158     2,158     0     0     0     0     0     0     1     0     0     0     0
0     0     0
AMRAMOS0 R             1         1,152     1,152     0     0     0     0     0     0     0     1     0     0     0
0     0     0
         S             1         1,152     1,152     0     0     0     0     0     0     0     1     0     0     0
0     0     0
ASMYTH0  R             5        11,701     2,340     0     0     0     0     0     0     2     3     0     0     0
0     0     0
         S             5        12,224     2,444     0     0     0     0     0     0     2     3     0     0     0
0     0     0
ASUTTON2 R             6        10,173     1,695     0     0     0     0     0     0     3     3     0     0     0
0     0     0
```

```
         S            6         15,447     2,574     0     0     0     0     0     0     3     3     0     0     0
0     0     0
ATFREVE0 R            3          4,762     1,587     0     0     0     0     0     0     3     0     0     0     0
0     0     0
AZC0GH00 S            4          8,632     2,158     0     0     0     0     0     0     0     4     0     0     0
0     0     0
BDUNKER0 R            7          9,821     1,403     0     0     0     0     0     0     6     1     0     0     0
0     0     0
BEWILLI0 R            3          9,372     3,124     0     0     0     0     0     0     2     1     0     0     0
0     0     0
         S            3         28,710     9,570     0     0     0     0     0     0     2     1     0     0     0
0     0     0
BMANZAN0 R            1          2,158     2,158     0     0     0     0     0     0     1     0     0     0     0
0     0     0
         S            1          2,158     2,158     0     0     0     0     0     0     1     0     0     0     0
0     0     0
BRASMUS0 R            3          6,474     2,158     0     0     0     0     0     0     3     0     0     0     0
0     0     0
         S            3          6,474     2,158     0     0     0     0     0     0     3     0     0     0     0
0     0     0
BRICHIN0 R            1          1,758     1,758     0     0     0     0     0     0     1     0     0     0     0
0     0     0
         S            1          3,807     3,807     0     0     0     0     0     0     1     0     0     0     0
0     0     0
BXSMI255 R            6         15,540     2,590     0     0     0     0     0     0     2     4     0     0     0
0     0     0
         S            6         15,540     2,590     0     0     0     0     0     0     2     4     0     0     0
0     0     0
CALLRVW0 R            2          1,988       994     0     0     0     0     0     0     0     2     0     0     0
0     0     0
         S            4         12,398     3,099     0     0     0     0     0     0     0     4     0     0     0
0     0     0
CBUNNEL0 R            2          3,454     1,727     0     0     0     0     0     0     0     2     0     0     0
0     0     0
CDESJAR0 R            5          8,229     1,645     0     0     0     0     0     0     2     3     0     0     0
0     0     0
CEWEBER1 R            4          6,768     1,692     0     0     0     0     0     0     0     4     0     0     0
0     0     0
CFERRAN4 R            1            503       503     0     0     0     0     0     0     1     0     0     0     0
0     0     0
CGALDE0  R            1          1,735     1,735     0     0     0     0     0     0     1     0     0     0     0
0     0     0
         S            1         86,750    86,750     0     0     0     0     0     0     1     0     0     0     0
0     0     0
CGALDE1  R            1          1,735     1,735     0     0     0     0     0     0     0     1     0     0     0
0     0     0
CGOOLSB1 R            2          2,926     1,463     0     0     0     0     0     0     0     2     0     0     0
0     0     0
CGUCINS3 R            9         19,371     2,152     0     0     0     0     0     0     8     1     0     0     0
0     0     0
         S            9         19,371     2,152     0     0     0     0     0     0     8     1     0     0     0
0     0     0
CKLARSO0 R            1            503       503     0     0     0     0     0     0     0     1     0     0     0
0     0     0
CMENO0   R            1          3,807     3,807     0     0     0     0     0     0     1     0     0     0     0
0     0     0
         S            1          3,807     3,807     0     0     0     0     0     0     1     0     0     0     0
0     0     0
COAAHS00 S           10         30,130     3,013     0     0     0     0     0     0     3     7     0     0     0
0     0     0
COJJHP00 S            4          8,632     2,158     0     0     0     0     0     0     4     0     0     0     0
0     0     0
COLAM030 R           20         31,330     1,566     0     0     0     0     0     0     9    11     0     0     0
0     0     0
CRUNNEL1 R            2          2,652     1,326     0     0     0     0     0     0     2     0     0     0     0
0     0     0
```

Note:

1. LINE RTN = Line Relative Terminal Number
2. R/S = Received/Sent

Messages—Queued But Not Sent (by transaction code)

```
     M E S S A G E S -- Q U E U E D   B U T   N O T   S E N T      D A T E
05/15/09            P A G E 00001
TRANSACTION          TOTAL
    CODE           MESSAGES
 VOMPCC1A                 1
 VOS1SIGN                 1
```

Messages—Program To Program (by transaction code)

```
     M E S S A G E S -- P R O G R A M   T O   P R O G R A M        D A T E
05/15/09            P A G E 00001
TRANSACTION          TOTAL
    CODE           MESSAGES
 VJDBI4T                  6
 VJDBL1T                144
 VJDBL3T                  9
 VJDBL4T                 88
 VJMPCC1                  2
 VJMPCC1A               117
 VJMPCC1B                 4
 VJMPCC1C                 1
 VJMPCC1D                 4
 VJMPCC1F                 1
 VJMPCC3                  2
 VOCITADD                 5
 VOCITFCT                 3
 VODSWT                  73
 VOGSC02                  3
 VOGSF05                  3
 VOGSJFX                  6
 VOMPCC1A               200
 VOMPCC1B                20
 VOMPCC2C                 1
 VOSSADD                 59
 VOSSCWL                  1
 VOSSESO                 11
 VOSSFDI                 20
 VOSSIA                  10
 VOSSNFH                 10
 VOSSNMA                 62
 VOSSSCR                  9
 VOSSSMI                  4
 VOSSSUP2                 1
 VOSSTIM                  1
 VOS019T1                30
 VOTABAD                  3
 VOTAFDI                  6
 VOTAFDO2                12
 VOTANSA                  1
 VOTAPAS                  2
 VOTARFT                  1
 VOTATREB                 1
 VOTATTE                  1
```

Transaction Report

```
     T R A N S A C T I O N   R E P O R T                    D A T E  05/15/09          P A G E 00001
TRANSACTION     TOTAL        TOTAL         AVG                HOURLY       DISTRIBUTION
   CODE  R/S   MESSAGES    CHARACTERS      SIZE  00-07 07-08 08-09 09-10 10-11 11-12 12-13 13-14 14-15 15-16 16-17
17-18 18-19 19-24
VGCAMSS  R           1         1,524     1,524     0     0     0     0     0     0     0     1     0     0     0
0     0     0
VGGDCIS  R           1         1,263     1,263     0     0     0     0     0     0     0     1     0     0     0
0     0     0
VGGDCMC  R           2         2,888     1,444     0     0     0     0     0     0     1     1     0     0     0
0     0     0
VGGDEWRA R           2         3,640     1,820     0     0     0     0     0     0     0     2     0     0     0
0     0     0
VGGDISS  R           4         2,136       534     0     0     0     0     0     0     3     1     0     0     0
0     0     0
VGGDRIP  R           4         6,728     1,682     0     0     0     0     0     0     1     3     0     0     0
0     0     0
VGGDRIS  R           1         1,718     1,718     0     0     0     0     0     0     1     0     0     0     0
0     0     0
VGGDRMC  R           4         6,868     1,717     0     0     0     0     0     0     3     1     0     0     0
0     0     0
VGGDSOIE R           5         8,316     1,663     0     0     0     0     0     0     2     3     0     0     0
0     0     0
VGGDTGM1 R           1         1,753     1,753     0     0     0     0     0     0     1     0     0     0     0
0     0     0
VGGDTGS1 R           1         1,718     1,718     0     0     0     0     0     0     1     0     0     0     0
```

```
0     0     0
VGGDTLOG R          47         62,741     1,334     0     0     0     0     0     0    24    23     0     0     0
0     0     0
VGGDTRAK R           5          7,960     1,592     0     0     0     0     0     0     5     0     0     0     0
0     0     0
VGG015T1 R           4          4,912     1,228     0     0     0     0     0     0     4     0     0     0     0
0     0     0
VGG022TC R           3          5,475     1,825     0     0     0     0     0     0     1     2     0     0     0
0     0     0
VGG022TE R           1          1,825     1,825     0     0     0     0     0     0     0     1     0     0     0
0     0     0
VGG022T1 R           7          5,879       839     0     0     0     0     0     0     3     4     0     0     0
0     0     0
VGG022T2 R           9         16,425     1,825     0     0     0     0     0     0     5     4     0     0     0
0     0     0
VGG043T1 R           7         10,789     1,541     0     0     0     0     0     0     3     4     0     0     0
0     0     0
VGS1SIGN R          13          3,289       253     0     0     0     0     0     0     6     7     0     0     0
0     0     0
VJCRLCRE R           1            674       674     0     0     0     0     0     0     1     0     0     0     0
0     0     0
         S           1            412       412     0     0     0     0     0     0     1     0     0     0     0
0     0     0
VJCRLREF R           1            681       681     0     0     0     0     0     0     1     0     0     0     0
0     0     0
         S           1          1,996     1,996     0     0     0     0     0     0     1     0     0     0     0
0     0     0
VJDBL4T  S           2          3,002     1,501     0     0     0     0     0     0     0     2     0     0     0
0     0     0
VJMPCC1  R           1          7,164     7,164     0     0     0     0     0     0     0     1     0     0     0
0     0     0
         S           1            576       576     0     0     0     0     0     0     1     0     0     0     0
0     0     0
VJMPCC1A R           7          1,381       197     0     0     0     0     0     0     0     7     0     0     0
0     0     0
VJMPCC1B S           1          3,850     3,850     0     0     0     0     0     0     0     1     0     0     0
0     0     0
VJMPCC1C S           3         24,497     8,165     0     0     0     0     0     0     0     3     0     0     0
0     0     0
VJMPCC1E S           2          1,994       997     0     0     0     0     0     0     2     0     0     0     0
0     0     0
VJMPCC2  R           6            822       137     0     0     0     0     0     0     5     1     0     0     0
0     0     0
         S           1            923       923     0     0     0     0     0     0     1     0     0     0     0
0     0     0
VJMPCC3  S           2            796       398     0     0     0     0     0     0     1     1     0     0     0
0     0     0
VJS1SIGN R           1            253       253     0     0     0     0     0     0     1     0     0     0     0
0     0     0
         S           1            253       253     0     0     0     0     0     0     1     0     0     0     0
0     0     0
VOCAMSS  R           6          9,144     1,524     0     0     0     0     0     0     5     1     0     0     0
0     0     0
         S          23         35,052     1,524     0     0     0     0     0     0    15     8     0     0     0
0     0     0
VOCICIQ1 R          61         16,403       268     0     0     0     0     0     0    31    30     0     0     0
0     0     0
         S          61        119,302     1,955     0     0     0     0     0     0    30    31     0     0     0
0     0     0
VOCICIQ2 R          11          4,785       435     0     0     0     0     0     0     5     6     0     0     0
0     0     0
         S          11         55,131     5,011     0     0     0     0     0     0     5     6     0     0     0
0     0     0
VOCIEAU1 R           1            248       248     0     0     0     0     0     0     1     0     0     0     0
0     0     0
         S           1            408       408     0     0     0     0     0     0     1     0     0     0     0
0     0     0
VOCIGARM R          22         18,843       856     0     0     0     0     0     0    12    10     0     0     0
0     0     0
         S          22          8,002       363     0     0     0     0     0     0    12    10     0     0     0
0     0     0
VOCIGIQ1 R          28          7,532       269     0     0     0     0     0     0    13    15     0     0     0
0     0     0
         S          28        424,865    15,173     0     0     0     0     0     0    13    15     0     0     0
0     0     0
VOCIMWL1 R           1            311       311     0     0     0     0     0     0     0     1     0     0     0
0     0     0
         S           1         63,211    63,211     0     0     0     0     0     0     0     1     0     0     0
0     0     0
VOCISRH1 R          30          8,278       275     0     0     0     0     0     0    16    14     0     0     0
0     0     0
         S          30         30,910     1,030     0     0     0     0     0     0    16    14     0     0     0
0     0     0
VOCITADD R           5          3,513       702     0     0     0     0     0     0     3     2     0     0     0
0     0     0
         S           5          7,659     1,531     0     0     0     0     0     0     3     2     0     0     0
0     0     0
VOCITFCT R           4          3,274       818     0     0     0     0     0     0     2     2     0     0     0
0     0     0
         S           4          8,082     2,020     0     0     0     0     0     0     2     2     0     0     0
0     0     0
VOCITIQ1 R          12          3,400       283     0     0     0     0     0     0     4     8     0     0     0
0     0     0
```

```
         S          12        26,918     2,243     0     0     0     0     0     0     4     8     0     0     0
0     0     0
VOCRLCRE R           1           674       674     0     0     0     0     0     0     0     1     0     0     0
0     0     0
         S           1           412       412     0     0     0     0     0     0     0     1     0     0     0
0     0     0
VOCRLREF R           1           681       681     0     0     0     0     0     0     0     1     0     0     0
0     0     0
         S           1         1,996     1,996     0     0     0     0     0     0     0     1     0     0     0
0     0     0
```

Transaction-Response Report

```
     T R A N S A C T I O N   R E S P O N S E   R E P O R T          D A T E  05/15/09           P A G E
00001
TRANSACTION      TOTAL          LONGEST          95%            75%            50%
25%          SHORTEST
   CODE        RESPONSES       RESPONSE       RESPONSE       RESPONSE       RESPONSE
RESPONSE       RESPONSE
  VJCRLCRE             1           .11S           .11S           .11S           .11S
.11S           .11S
                       1           .08S           .08S           .08S           .08S
.08S           .08S
  VJCRLREF             1           .03S           .03S           .03S           .03S
.03S           .03S
                       1           .03S           .03S           .03S           .03S
.03S           .03S
  VJMPCC1A             9          5.02S          5.01S          5.01S           .29S
.05S           .04S
                       9           .36S           .31S           .16S           .06S
.03S           .01S
  VJMPCC1C             1           .10S           .10S           .10S           .10S
.10S           .10S
                       1           .02S           .02S           .02S           .02S
.02S           .02S
  VJMPCC1E             1           .01S           .01S           .01S           .01S
.01S           .01S
                       1           .01S           .01S           .01S           .01S
.01S           .01S
  VJMPCC2              1           .01S           .01S           .01S           .01S
.01S           .01S
                       1           .00S           .00S           .00S           .00S
.00S           .00S
  VJS1SIGN             1           .18S           .18S           .18S           .18S
.18S           .18S
                       1           .00S           .00S           .00S           .00S
.00S           .00S
  VOCAMSS             23           .06S           .03S           .02S           .01S
.01S           .00S
                      23           .06S           .03S           .02S           .01S
.00S           .00S
  VOCICIQ1            61           .42S           .36S           .17S           .05S
.01S           .00S
                      61           .42S           .36S           .17S           .05S
.01S           .00S
  VOCICIQ2            11           .03S           .03S           .03S           .02S
.01S           .01S
                      11           .03S           .03S           .03S           .02S
.01S           .01S
  VOCIEAU1             1           .03S           .03S           .03S           .03S
.03S           .03S
                       1           .03S           .03S           .03S           .03S
.03S           .03S
  VOCIGARM            22           .31S           .01S           .00S           .00S
.00S           .00S
                      22           .31S           .01S           .00S           .00S
.00S           .00S
  VOCIGIQ1            28           .20S           .02S           .02S           .01S
.00S           .00S
                      28           .20S           .02S           .02S           .01S
.00S           .00S
  VOCIMWL1             1           .09S           .09S           .09S           .09S
.09S           .09S
                       1           .09S           .09S           .09S           .09S
.09S           .09S
  VOCISRH1            30           .18S           .14S           .09S           .05S
.00S           .00S
                      30           .18S           .14S           .09S           .05S
.00S           .00S
```

```
  VOCITADD             5           .11S           .10S           .10S           .09S
.08S           .06S
                     5           .11S           .10S           .10S           .09S
.08S           .06S
  VOCITFCT             4           .10S           .06S           .06S           .06S
.05S           .05S
                     4           .10S           .06S           .06S           .05S
.05S           .05S
  VOCITIQ1            12           .17S           .15S           .12S           .10S
.02S           .00S
                    12           .17S           .15S           .12S           .10S
.02S           .00S
  VOCRLCRE             1           .18S           .18S           .18S           .18S
.18S           .18S
                     1           .08S           .08S           .08S           .08S
.08S           .08S
  VOCRLREF             1           .03S           .03S           .03S           .03S
.03S           .03S
                     1           .00S           .00S           .00S           .00S
.00S           .00S
  VODSWD             132           .21S           .04S           .01S           .00S
.00S           .00S
                   132           .20S           .03S           .00S           .00S
.00S           .00S
  VOGSC01              6           .31S           .23S           .21S           .17S
.11S           .06S
                     6           .31S           .23S           .21S           .17S
.11S           .05S
  VOGSC02              3          1.05S           .97S           .97S           .97S
.75S           .75S
                     3           .96S           .74S           .74S           .74S
.48S           .48S
  VOGSF05             26           .25S           .08S           .02S           .01S
.01S           .00S
                    26           .24S           .07S           .01S           .01S
.00S           .00S
  VOGSF07            151         26.69S          1.13S           .04S           .02S
.01S           .00S
                   151         12.27S           .04S           .03S           .02S
.01S           .00S
  VOGSF10              1           .07S           .07S           .07S           .07S
.07S           .07S
                     1           .03S           .03S           .03S           .03S
.03S           .03S
  VOGSF11             91           .17S           .06S           .02S           .01S
.01S           .00S
                    91           .16S           .04S           .01S           .01S
.00S           .00S
  VOGSJFX              6         43.73S         23.72S          4.02S           .03S
.01S           .01S
                     6         41.13S           .09S           .05S           .03S
.01S           .00S
  VOMPCC1              1           .00S           .00S           .00S           .00S
.00S           .00S
                     1           .00S           .00S           .00S           .00S
.00S           .00S
  VOMPCC1A            87          6.32S           .39S           .05S           .03S
.02S           .01S
                    87          6.32S           .34S           .04S           .03S
.02S           .00S
  VOMPCC1B             7           .38S           .10S           .09S           .01S
.00S           .00S
                     7           .10S           .10S           .09S           .00S
.00S           .00S
```

Application Accounting Report

```
     A P P L I C A T I O N   A C C O U N T I N G   R E P O R T      D A T E  05/15/09           P A G E 00002
PROGRAM TRANSACTION    MESSAGE- - - - COUNTS    DATA - - - - - - - - - - BASE - - - - - - - - - COUNTS  CC OR RC    TOT
PROG      AVG
 NAME      CODE    PRI  QTY    GU    GN  ISRT    GU    GN   GNP   GHU   GHN  GHNP  ISRT  DLET  REPL   NOT 0      CPU
TIME      TIME
VOSSBIR  VOSSBIR   8     4     6     0     4    15     0     0     7     0     3     7     3     0     0
0.0S      0.002S
VOSSCHC  VOSSCHC   9     9    13     0     6    54     9     0    19     0     7    18     8     0     0
0.0S      0.003S
VOSSCWL  VOSSCWL   8     1     2     0     3    29     8     0    16     0     0     5     1     6     0
0.0S      0.019S
VOSSCWLR VOSSCWLR  8     2     3     0     2    10    16     0     4     0     0     2     2     0     0
```

```
0.0S      0.002S
VOSSEMAR VOSSEMAR  8     2     3     0     6    20     0     0     4     0     2    10     2     0     0
0.0S      0.005S
VOSSESO  VOSSESO   8    20    26     0    24   126     0     0    42     0     0     1     0    35     0
0.0S      0.002S
VOSSFDI  VOSSFDI   8    17    20    34    37    25    10     0    26     2     0    16     0     4     0
0.0S      0.002S
VOSSGIU  VOSSGIU   8     1     2     0     6    91    94    20    37     0     0    10     1    24     0
0.0S      0.034S
VOSSGSR  VOSSGSR   8     1     2     0    64    89 1,087     0     3     0     1    64     1     0     0
0.0S      0.044S
VOSSIAP  VOSSIA    9    46    54     0    57   732    34    57   394    16    51   134    53   184     0
0.3S      0.007S
VOSSIHP  VOSSIH    9     1     2     0     0     2     0     0     1     0     0     0     1     0     0
0.0S      0.002S
VOSSIHR  VOSSIHR   8     2     4     2     4     5    16     0     9     0     0     2     1     2     0
0.0S      0.007S
VOSSLOG  VOSSLOG   6     2     3     0     0     0     0     0     2     0     0     2     0     0     0
0.0S      0.001S
VOSSLRKR VOSSLRKR  8    74    77     0    74   295   438     0   581    73     1   148    74     0     0
0.2S      0.002S
VOSSNFH  VOSSNFH   8    38    44     0    31   182    11    95   192    72    39   128    79    72     0
0.2S      0.007S
VOSSNMA  VOSSNMA   8    28    35    56    84     0     0     0     0     0     0     0     0     0     0
0.0S      0.001S
VOSSSCR  VOSSSCR   8    10    15    54    60   160    19    20   109     0    10    55     5    47     0
0.1S      0.015S
VOSSSMI  VOSSSMI   8     2     4     8    10     0     0     0     0     0     0     0     0     0     0
0.0S      0.004S
VOSSSUP  VOSSSUP2  8     1     2     0     2     4     1     0    10     2     0     7     1     4     0
0.0S      0.016S
VOSSTDCP VOSSTDC   6     1     1     0     0     0     0     0     1     0     0     0     0     0     0
0.0S      0.000S
VOSSTIM  VOSSTIM   8     1     2     0     2     5     0     0     2     0     0     3     0     1     0
0.0S      0.011S
VOS008B1 BMP/JBP   0     0     0     0     0    69 4,198     0    27 2.4E4     0     0     0   214     0
0.0S      0.000S
VOS019T  VOS019T1  9    36    44     0    71   759    32    22   401   268   220   421   373   168     0
0.5S      0.016S
VOS027T  VOS027T1  9   207   211     0   207   851     0     0   248     0    44   414   196     0     0
0.2S      0.001S
VOS033T  VOS033T1  9     8    12     0     8   212    73    62    85     4     4    16     5     0     0
0.1S      0.013S
VOS038T  VOS038T1  9    39    44     0    39   157     0     0    43     0     4    39    38     1     0
0.0S      0.001S
VOS039T  VOS039T1  9   103   105     0   188   778 3,611    81   121     0    36   565   103     0     0
0.4S      0.003S
VOS041T  VOS041T1  8     3     6     0     4    18     0     9     7     0     2     9     3     0     0
0.0S      0.005S
VOS042T  VOS042T1  9   214   217     0   214 1,409   158    25   241     0    28   428   203     0     0
0.3S      0.001S
VOS043T  VOS043T1  9    52    54     0   194 5,390 1.6E4    57   100     0    48   388    51     0     0
2.0S      0.038S
VOS1SGN  VOS1SIGN 10    23    28     0    43    85   788     0    40     0     0     0     0    40     0
0.0S      0.002S
VOS102T  VOS102T1  8     4     7     0     4    24     0     0    12     0     0     6     4     4     0
0.0S      0.004S
VOS104T  VOS104T1  8    16    22     0    12    69     0     0    28     0     0    12    12     0     0
0.0S      0.002S
VOTAACTR VOTAACTR  8     7    12     0     7    28     0     0     7     0     0     7     7     0     0
0.0S      0.003S
VOTABAD  VOTABAD   9     8    14     0     4   356    84   152   234    52    18   288    51    61     0
0.2S      0.031S
VOTADRP  VOTADRPA  8     2     3     0     1     2 2,504     0     0     0     0     0     0     0     0
0.2S      0.129S
VOTAFDI  VOTAFDI   8     3     4     2    10     7     0     2     9     6     1    10     6     2     0
0.0S      0.006S
VOTAFDO  VOTAFDO2  8     8    13    17    23    51     5     4    75    46    10    60    47    23     0
0.1S      0.013S
VOTAFEB  VOTAFEB   8     2     3     0     2    10     0     0     4     0     1     3     2     1     0
0.0S      0.005S
VOTAMDS  VOTAMDS   8    12    20     0     0     0     0     0     0     0     0     0     0     0     0
0.0S      0.001S
VOTANSA  VOTANSA   8     1     2     0     1     2     0     1     3     0     0     7     1     0     0
0.0S      0.015S
VOTAPAS  VOTAPAS   9     2     4     0     3     8     0     1    11    10     2    17    10     3     0
0.0S      0.019S
VOTAPTA  VOTAPTA   8     2     4     4     0     0     0     0     2     0     0     0     2     0     0
0.0S      0.002S
VOTARFT  VOTARFT   8     2     4     7     4     7     0     0    11     6     2    11     4     2     0
0.0S      0.013S
VOTASEB  VOTASEB   8    10    14     0    10    15     0     0     0     0     0     0     0     0     0
0.0S      0.001S
VOTATBL  VOTATTE  12     1     2     0     2    38     5     3    10     0     1    19     1     2     0
0.0S      0.023S
VOTATEB  VOTATREB 12     3     4     0     4    14     0     0     9     4     3     8     7     1     0
0.0S      0.004S
VOTATSR  VOTATSR   8     1     2     0     2     5     0     0     5     2     1     3     0     0     0
0.0S      0.004S
VOTATST  VOTATST   8     1     2     0     0     2     0     0     3     0     3     0     1     0     0
0.0S      0.003S
VOXPECI  VOXPECIN  8     1     2     0     0     0     0     0     0     0     0     0     0     0     0
0.0S      0.013S
```

```
SYSTEM TOTALS       3,097 3,431 3,238 7,999 2.5E4 3.7E4 1.9E4 1.0E4 2.5E4   945 1.2E4 2,964 2,730     0
14.0S      0.004S
       I M S   ACCOUNTING   REPORT                                    D A T E  05/15/09                    P A G
E 00001


         START TIME 19:59:34
          STOP TIME 20:00:22
REPORT PERIOD IS FROM 05/15/09 TO 05/15/09.


END OF REPORTS
```

*****

Second insert is counted for single user issued insert if all the following conditions are met:

1. New HIDAM or PHIDAM Root
2. Not Duplicate Key (II status not returned)

******

These dates will not appear unless the input to DFSIST30 is sorted with date control.

Message Select and Copy or List option report

The following figure shows an example of the report produced by the Message Select and Copy or List option.

```
            MESSAGES

INPUT    TRANSACTION  LINE   RELA  SEQ  SYMBOLIC               OUTPUT  NODE      SEQ    SYMBOLIC
PREFIX        CODE     NO    TERM  NO   ADDRESS   DATE    TIME  PREFIX  NAME      NO     ADDRESS
DATE    TIME
        THIS OUTPUT NOT RESULT OF INPUT                                  DSWP5008  00017  PDSW5008
09.107  15.54.1
                     3
OUTPUT  SEG=001 LEN=0001*F*
INPUT    TRANSACTION  LINE  RELA   SEQ  SYMBOLIC               OUTPUT  NODE      SEQ    SYMBOLIC
PREFIX        CODE     NO   TERM   NO   ADDRESS   DATE    TIME  PREFIX  NAME      NO     ADDRESS
DATE    TIME
        THIS OUTPUT NOT RESULT OF INPUT                                  DSWP5008  00019  PDSW5008
09.107  15.54.4
OUTPUT SEG=001 LEN=0009*88-3-2000*
                                        3              3                 3                          3
OUTPUT SEG=002 LEN=0248*WITHDRAWAL   $300.00 FDEPOSIT   $6704.62 FSAVINGS   444.44 FCHECKING    $9800.50
F*
                                        3              3                 3                          3
                      *OVERDRAFT     $30.32FVISA       $2020.20 FMASTER CHRGE $105.00 FCARLOAN $1040.00
F*
                                       3
                      *TRANSFER C-5  $50.00 FCHRISTMAS CLUB  $94.60*
INPUT    TRANSACTION    NODE       SEQ  SYMBOLIC                  OUTPUT  NODE      SEQ    SYMBOLIC
PREFIX      CODE        NAME       NO   ADDRESS   DATE       TIME  PREFIX  NAME      NO     ADDRESS
DATE    TIME
          DE1Q          DSWPOO56  00015 PDSW0056  09.107   15.53.51        DSWP0056  00016  PDSW0056
07.107  15.54.2
                        03     0  18D
INPUT  SEG=001 LEN=0016*1BDE1Q  3Y43A*
INPUT  SEG=002 LEN=0230* 23(9) WITHDRAW OF $300DEPOSIT OF $6704.62SAVINGS DEPOSIT OF $444.44CHECKING
TRANSFER OF $9800.50OV*
                      *ERDRAFT OF $30.32VISA ENTRY OF $2020.20MASTER CHARGE OF $105.00CAR LOAN OF
$140.00TRANSFER C-S OF $*[1]
                      *50.00CHRISTMAS CLUB OF $94.60Y
INPUT    TRANSACTION    NODE       SEQ  SYMBOLIC
PREFIX      CODE        NAME       NO   ADDRESS   DATE       TIME
          DE1Q          DSWP0084  00017 PDSW5008  09.107   15.54.25
                                                  3
OUTPUT SEG=001 LEN=0031*+ DATA SUCCESSFULLY RECEIVED +F*
INPUT    TRANSACTION  LINE   RELA  SEQ  SYMBOLIC                  OUTPUT   NODE      SEQ    SYMBOLIC
PREFIX      CODE       NO    TERM  NO   ADDRESS   DATE       TIME  PREFIX   NAME      NO     ADDRESS
DATE    TIME
          DE1Q[2]      DSWP016    00018 PDSW0116  09.107   15.54.36         DSWP0116  00016  PDSW0116
07.107  15.54.36
```

Notes:

1. Indicates a 230-character report message
2. Indicates a 31-character message generated by the transaction code "DE1Q" and transmitted to a relative terminal DSWP0116.

Part 4. Log utilities

Use the log utilities to produce an SLDS, to generate a data set from system log data sets, and to generate a usable log data set from a log data set that contains errors or was not properly terminated.

Each topic introduces how the utility works, defines requirements and restrictions for its use, and provides examples.

© Copyright IBM Corp. 1974, 2024

Chapter 21. Log Archive utility (DFSUARC0)

You can use the Log Archive utility (DFSUARC0) to produce an SLDS from a filled OLDS or a batch IMS SLDS.

IMS DB writes log records on an SLDS that can be on tape or DASD. This allows an IMS batch user to log to DASD, create an SLDS, and later copy that SLDS to DASD or tape.

The Log Archive utility provides the following optional functions. You must specify these functions with utility control statements.

Creating an RLDS (recovery log data set)

You can request creation of an output data set containing all the log records needed for DB recovery. The output data set is referred to as a recovery log data set (RLDS). If the input data set contains records for DB recovery, the RLDS is known to DBRC and is used in place of the SLDS by GENJCL when creating JCL for DB recovery and change accumulation. If the input data set contains no records needed for DB recovery, the RLDS is a null data set. In this case DBRC records the data set name and volume serial number of the SLDS, in place of the RLDS DSNAME and volume serial number, and then uses the SLDS for GENJCL instead of the null RLDS.

Omitting log records on an SLDS

Generally, the SLDS should contain all the log records from the OLDS, but if you need to omit some types of log records from the SLDS, these log records must be specified in an SLDS control statement, using the NOLOG parameter. The SLDS must contain those records that might be needed for database recovery and IMS restart. The Log Archive utility will issue an error message and terminate if a required record type is specified to be omitted.

Copying log records into user data sets
The Log Archive utility can copy selected log records into multiple user data sets directly. In SYSIN control statements, you can specify the log records to be selected and the ddname of the data set to which the records are to be written.

Specifying user exit routines
You can specify multiple user exit routines for the archive utility. The Log Archive utility passes control to each user exit routine at initialization, input log read, and termination time. User exit routines can process the log records or create a data set.

Specifying forced end of volume (EOV)
To ensure that corresponding volumes in a dual SLDS on tape contain the same records (and consequently are interchangeable), the number of blocks to be written on a volume can be specified. EOV will be forced to both SLDSs when the specified number of log blocks have been written.

Subsections:

- “Restrictions” on page 494
- “Prerequisites” on page 494
- “Requirements” on page 494
- “Recommendations” on page 494
- “Input and output” on page 494
- “JCL specifications” on page 496
- “Utility control statements” on page 497
- “Return codes” on page 501

© Copyright IBM Corp. 1974, 2024

Restrictions

Currently, no restrictions are documented for the DFSUARC0 utility.

Prerequisites

Currently, no prerequisites are documented for the DFSUARC0 utility.

Requirements

Currently, no requirements are documented for the DFSUARC0 utility.

Recommendations

Currently, no recommendations are documented for the DFSUARC0 utility.

Input and output

The Log Archive utility has two types of input: OLDSs and SLDSs. The utility accepts only log data sets that are created by the same release of IMS as the utility release level.

OLDS input

The OLDS used for input must have been successfully closed. The status in RECON for the input OLDS must be 'ARCHIVE NEEDED'.

An error in a single OLDS causes the archive job to terminate. Run the Log Recovery utility to recover the OLDS, and rerun the Log Archive utility.

If dual OLDSs were used during IMS online execution, both are used as input to the Log Archive utility. If an error is encountered in the primary OLDS, the archive utility switches to the secondary OLDS. If the record is found in the secondary OLDS, the archive job continues. If an error is encountered in the same block, the archive job terminates. Run the Log Recovery utility to recover the OLDS, and rerun the Log Archive utility. If one dual OLDS is not available, for example the status is not 'ARCHIVE NEEDED', only the available OLDS is used as input. The unavailable OLDS is ignored.

If dual OLDSs are used as input and an error exists in the first block of the primary OLDS, the Log Archive utility terminates unsuccessfully. Sequence errors are indicated on the first block of both OLDSs, even though the secondary OLDS might be correct. The Log Archive utility uses the first block of the primary OLDS as an anchoring point. If this block is in error, data collected from it cannot be verified by comparison to the secondary OLDS. If errors exist on the first block of either OLDS, run the Log Recovery utility to recover the OLDS, then rerun the Log Archive utility.

If multiple OLDSs are specified as the input OLDS, they must have been created consecutively. OLDSs created by different IMS system executions cannot be input at one time.

If any OLDS in the input was terminated at a recovery point (a recovery point results at every **/DBRECOVERY** and **/DBDUMP** command that forces an OLDS switch), the archive utility performs as follows:

- If at least one of the SLDSs and RLDSs is placed on DASD, the output data sets are closed and the archive job terminates after processing any OLDS that terminates at a recovery point. Remaining OLDS that might not have been processed are still in a state of ARCHIVE NEEDED.
- If all SLDSs and RLDSs are placed on tape, IMS forces end of volume for all SLDSs and RLDSs and the archive job continues using the next volume for the SLDS and RLDS.

DBRC verifies the input OLDS. If there is an error in the OLDS specifications, the Log Archive utility terminates with an error message.

SLDS input

The single SLDS used for input is the SLDS on DASD created in an IMS batch environment. Also, this SLDS must have closed successfully. When DBRC=NO is specified in the EXEC parameter, tape SLDS input is permitted. You can use the SLDS of a previous archive and archive it again; however, this is not an intended use.

Dual SLDSs can be used as input. If an error is encountered in the primary SLDS, the Log Archive utility switches to the secondary SLDS. If the record is found on the secondary SLDS, the archive job continues. An error in a single SLDS or errors in the same block in dual SLDSs terminates the archive job. Run the Log Recovery utility to recover the SLDS and rerun the Log Archive utility.

When the input is from a DASD SLDS created with DBRC present, the Log Archive utility will notify DBRC to update the existing SLDS record with the new SLDS information. You must create the JCL for the archive job of a batch SLDS.

Output

In addition to the SLDS, the optional RLDS, and the user data set produced as output, the Log Archive utility also produces a listing in SYSPRINT. SYSPRINT contains the following:

- A listing of control statements
- A listing of checkpoint time stamp IDs
- A listing of descriptive messages
- A listing of the result of the archive

The following figure is an example of a SYSPRINT listing of control statements.

```
********LOG ARCHIVE UTILITY CONTROL STATEMENT********
SLDS -
   NOLOG(10,16,5F,67,69) FEOV(08000)
COPY  DDNOUT1(DATASET1) -
  RECORD(OFFSET(5) FLDTYP(X) VALUE(16) FLDLEN(1) COND(E)) -
  RECORD(OFFSET(5) FLDTYP(X) VALUE(50) FLDLEN(1) COND(E)) -
  RECORD(OFFSET(5) FLDTYP(X) VALUE(51) FLDLEN(1) COND(E)) -
  RECORD(OFFSET(5) FLDTYP(X) VALUE(52) FLDLEN(1) COND(E))
EXIT  NAME(UEXIT01)
```

The following figure is an example of a listing of checkpoint time stamp IDs.

```
  USER CHECKPOINT RECORD - yyyy.ddd hh:mm:ss.t CHKPT-id region-id prg-name (v1)(v2)
SYSTEM CHECKPOINT RECORD - yyyy.ddd hh:mm:ss.t chkpt-id (v1)(v2) CHECKPOINT XXX  (RESTART TTT)
```

When checkpoint log records (X'18' and X'4001') are found, the SYSPRINT listing prints one of the preceding output lines. Date, time, and checkpoint ID are shown for both. Region-ID and program name are for X'18' records; checkpoint request type is for X'4001' records, where XXX is the type of checkpoint requested in English. Also shown is the volume serial of the output primary SLDS volume (v1) and, if dual output, the secondary SLDS volume (v2) to which the checkpoint is copied. Restart type is also given for the first X'4001' record where TTT is the type of restart performed in English.

The following figure is an example of a SYSPRINT listing of descriptive messages.

```
*** END OF VOLUME FORCED ON SLDS. PRIMARY(volser) SECONDARY(volser) ***
*** WRITE ERROR ON SLDS|USER|RLDS ddname ***
*** OUT-OF-SPACE on SLDS|USER|RLDS ddname ***
*** NO RECORD FOUND FOR SLDS|USER|RLDS ddname ***
```

The following figure shows an example of a SYSPRINT listing of the result of the archive.

```
*** LOG ARCHIVE UTILITY (DFSUARC0)               **hh:mm yy.ddd **
       COPIED LOG RECORDS

FROM     DDNAME=ddname VOLSER=volser          DDNAME=ddname VOLSER=volser
```

```
           (for primary input)                    (for secondary input)
                 .
                 .       .
TO PRIMARY SLDS DSNAME=dsname
    VOLSER =   volser  volser  volser .............
TO SECONDARY SLDS DSNAME=dsname
    VOLSER =   volser  volser  volser .............

SLDS DOES NOT CONTAIN THE FOLLOWING LOG RECORDS:
  'xx' 'xx' 'xx' 'xx' ......

TO PRIMARY RLDS   DSNAME=dsname
     VOLSER =   volser  volser  volser .............
TO SECONDARY RLDS DSNAME=dsname
     VOLSER =   volser  volser  volser .............
```

JCL specifications

The Log Archive utility runs as a z/OS batch job, and multiple log archive utility jobs can execute concurrently. A job statement, an EXEC statement, and DD statements that define input and output are required. When dual output is requested, the SLDS consists of primary and secondary data sets.

EXEC statement

Defines the utility to be run and optional execution parameters. Its format is:

```
//STEP EXEC PGM=DFSUARC0
           PARM= 'nnnn, DBRC=nnn, IMSPLEX=imsplex_name, DBRCGRP=xxx'
```

PARM=
Indicates the subsystem identifier and whether DBRC is specified.

nnnn
Indicates a 1- to 4-character IMS subsystem identifier and must be specified if the input data set is an OLDS. This is the same value as the IMSID for the online IMS system that created the data in the OLDS.

DBRC=YES|NO
DBRC=NO (or N) can be specified to explicitly declare that DBRC is not to be used for this execution of this utility.

DBRC=YES (or Y) can be specified to explicitly declare that DBRC is to be used for the execution of this utility.

If DBRC= is not specified, YES is the default.

IMSPLEX=*imsplex_name*
Indicates which IMSplex DBRC should join. IMSPLEX= is an optional parameter.

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

DD statements

STEPLIB DD
Points to the program libraries that contains the Log Archive program and to any user exit routines.

DFSOLPnn DD (for primary OLDS)
DFSOLSnn DD (for secondary OLDS)
Describes the OLDS used for input. You can specify dual OLDSs. In the case of dual OLDSs, the suffixes of the primary and secondary OLDS must match. The value of *nn* (the suffix) is 00 through 99 and must be the same *ddname* that was used when the log data was created by online execution. All OLDSs used as input must have been used consecutively during an online execution.

If DBRC=Y, the OLDS DD statements can be specified in any sequence in the DD statements.

If DBRC=N, the OLDS DD statements must be specified in the sequence they were created, and dual OLDS must be specified as a sequence of primary and secondary pairs.

You can specify between 2 and 99 read buffers for the DCB BUFNO keyword.

DFSSLDSP DD (for primary input SLDS)
DFSSLDSS DD (for secondary input SLDS)
Specifies the batch SLDS. Optionally, you can specify a dual SLDS for a batch SLDS. A SLDS and an OLDS used for input are mutually exclusive. You can specify 2 through 99 read buffers.

DFSSLOGP DD (for primary output SLDS)
DFSSLOGS DD (for secondary output SLDS)
Defines the SLDS used for output. Its format will depend on the device type used. If the SLDS is on DASD, you must allocate sufficient space to contain the log being archived. The SLDS block size can be specified and can be different from the input data set block size. If not specified, the block size of the input data set is used. The secondary SLDS is optional and specifies dual archiving. If the input is a batch SLDS and the Log Archive utility is run with DBRC present, dual output can be created only if dual SLDS records are already known to DBRC.

If dual SLDSs are being created, they can have different block sizes. However, if FEOV is specified, it is ignored unless the block size of both data sets are equal and both are allocated to tape. If tape is specified, it must have a standard label. You can specify 2 through 99 write buffers.

Restriction: Do not use the JCL parameter FREE=CLOSE on these DD statements. The data sets are dynamically deallocated, and using FREE=CLOSE can produce unpredictable results.

ddname DD (for either RLDS or user output data set, or both)
Defines either a user data set or recovery log data set (RLDS) or both. If the data set is on DASD, you must allocate sufficient space to contain the records being copied to it. The data set is created with RECFM=VB. The block size can be specified and can be different from the block size of the input data set, but it must be large enough to contain your longest record. If not specified, the block size of the input data set is used. If dual data sets are being created, they can have different block sizes. You can specify 2 through 99 write buffers.

SYSPRINT
Defines the output message data set.

SYSUDUMP
Defines the dump data set.

SYSIN DD
Specifies the control statements.

RECON1 DD
Defines the first DBRC (Database Recovery Control) RECON data set. This RECON1 data set must be the same RECON1 data set used by the IMS control region.

RECON2 DD
Defines the second DBRC RECON data set. This RECON2 data set must be the same RECON2 data set used by the IMS control region.

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

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

Utility control statements

All control statements are optional. Use the control statements when:

- Using user exit routines
- Creating an RLDS
- Placing certain records into a user data set

- Eliminating certain records from being copied to the SLDS
- Forcing duplicate tape output volumes

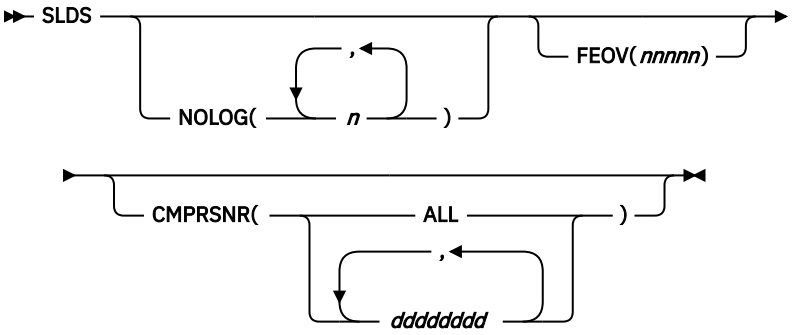
There are three types of control statements, and each statement consists of an operation code and parameters. The rules for using the control statement are:

- Control statements can be placed in columns 1 to 72 in free format. Parameters can be in any sequence.
- Each operation code and parameter must be separated with a blank, a comma, or a comment.
- Multiple lines can be used for a control statement. Continuation characters (+ and -) can be placed between columns 1 and 72. If (+) is used, the lines are concatenated without a blank. If (-) is used, the lines are concatenated with a blank.
- The value of any parameter must be specified between single parentheses.

SLDS statement

An SLDS statement specifies log record types that are not written to the SLDS. It also specifies that end-of-volume is forced for tape output volumes. If omitted, all log records are copied to the SLDS. Only one SLDS control statement is allowed.

The format of the SLDS control statement is:



NOLOG
Defines the log record types that are not to be copied to the SLDS. The value of a NOLOG subparameter should be specified in hexadecimal, for example, SLDS NOLOG (19,1A,1B).

The SLDS must contain those records that might be needed for database recovery and for system restart. The Log Archive utility issues an error message and terminates if a required record type is specified to be omitted.

FEOV
Specifies duplicate output tape volumes. This parameter is only applicable in a dual tape SLDS environment. It ensures that corresponding volumes in a multivolume data set contain the same records (and consequently are interchangeable).

nnnnn indicates the number of blocks to be written to a tape SLDS. Each time the blocks are written, a FEOV is issued for both the primary and secondary SLDSs. The block number is specified in 5 decimal digits. If the block sizes of both SLDSs are not equal, the FEOV parameter is ignored.

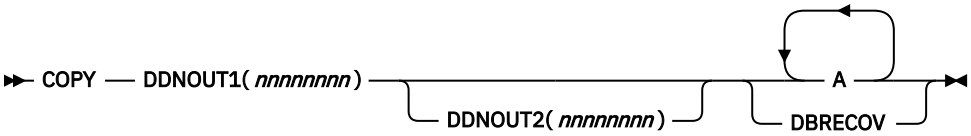
CMPRSNR
Determines the disposition of the database update log records for nonrecoverable full-function databases. Specifying CMPRSNR overrides the default behavior of copying the log records in to the archive.

CMPRSNR(ALL)
Indicates that the log records are compressed. Compressing the records causes them to be replaced by minimal size placeholder records.

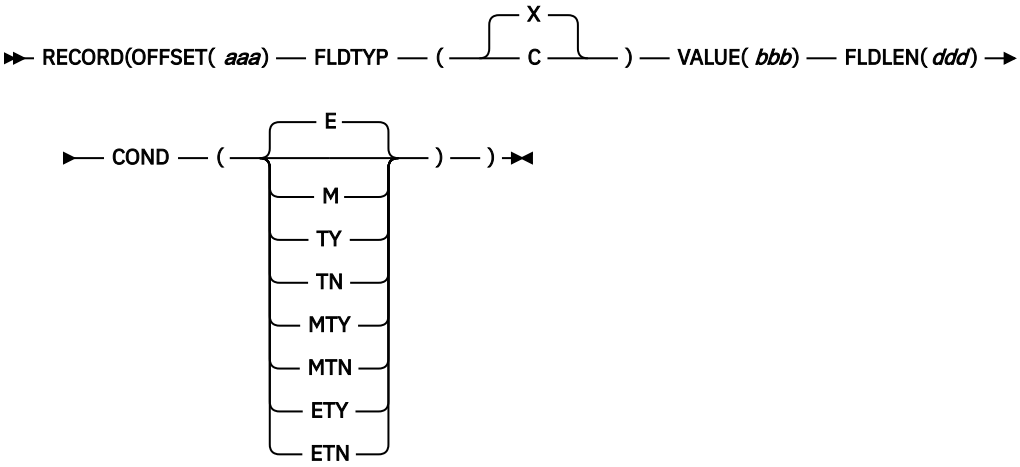
CMPRSNR(*dddddddd*)
An 8-character string that indicates that the log records are compressed only if the database name matches the *dddddddd* string.

COPY statement

The COPY statement is used to create an RLDS or a user data set during archive. The format for the COPY statement is:



A



The following abbreviations can be used in place of the keywords in the COPY statement:

Keyword
Abbreviation

OFFSET
O

FLDTYP
T

VALUE
V

FLDLEN
L

COND
C

DDNOUT1
DDNOUT2
Identifies the ddnames of the data sets. DDNOUT2 only applies if dual copies are being created. The DD statements must be included in the JCL. *nnnnnnnn* is a ddname value.

RECORD
Identifies the conditions for selecting a record to be written to the specified data set.

OFFSET(*aaa*)
Defines the beginning of the field to be tested in the record. The default is position one of the record.

aaa is the value and can be in the range from 1 up to and including the length of the record under test. Maximum value is 32767 bytes. No checking is performed to determine if the logical record length is exceeded. The value specified in the OFFSET keyword is always expressed as relative to byte 1.

FLDTYP(X)|(C)
Defines the type of data in the VALUE field. A value of X or C must be specified.

X defines the data to be treated as hexadecimal character pairs. The test data is packed, two bytes into one, to form hexadecimal equivalents. X is the default.

C defines the data to be treated as EBCDIC.

VALUE(*bbb*)
Can be specified in hexadecimal if FLDTYP(X) is specified, or in EBCDIC if FLDTYP(C) is specified. The value is specified between quotation marks in EBCDIC. The quotation mark notation is required when the character string contains a separator of blank or comma. Any characters can be specified within the quotation marks. (Double quotation marks within quotation marks represent a single quotation mark.) If a minus sign is the last nonblank character, it is assumed that the value is continued on the next line.

Restriction: The value of *bbb* cannot exceed 255 EBCDIC or 510 hexadecimal characters.

The length of this field is determined by the FLDLEN value and not by the number of "nonnull" characters in this field.

FLDLEN(*ddd*)
Defines the number of characters to be used from the test field.

ddd represents the actual number of bytes to be used, not the number of characters specified in VALUE. The acceptable range of values for this field is 1 to and including 255. The default is 1.

COND(*x*)
Defines the type of test and its relationship to other tests in the group. The default is COND(E). You can specify COND(*x*) singularly or join multiple options together.

E
Marks the last (or only) element in a test series. Any record control statements appearing after this form a new series of tests. This allows various tests to be performed on each record and each test series can be used on different fields within the record.

M
Indicates this is a multifield test; more than one test is to be made on each input record. All tests in this series must be satisfied before final output selection and processing of this record can begin.

T
Causes the VALUE byte to be used as a "test under mask" value, instead of a compare field. Only the first byte (two hexadecimal characters if FLDTYP(X)) of the VALUE field will be used. If FLDTYP(C) is used, the hexadecimal equivalent of the EBCDIC character is the test value. If this parameter is used, the FLDLEN keyword must not be specified and a default length of one is assumed.

Y
Indicates that there must be a bit in the record test field for each corresponding bit of the test byte for the "test under mask." This is equivalent to a "branch if ones" test.

N
Indicates that there must not be a bit in the record test field for any of the corresponding bits of the test byte for the "test under mask." This is equivalent to a "branch if zeros" test.

MT
Defines a "test under mask" option with the properties of a multifield test. This parameter must be used for a multifield test that starts with a "test under mask" value.

ET
Signifies that a multifield test series ends with a "test under mask" condition.

DBRECOV
Copies all log records needed for database recovery to the specified output data set. This output data set is known to DBRC and is used by the GENJCL process in lieu of the created SLDS when creating JCL for DB Recovery or Change Accumulation. This output data set is the recovery log data set (RLDS). If there are no records needed for DB recovery, the RLDS is a null data set. In this case DBRC records the DSNAME and volume serial number of the SLDS, in place of the RLDS DSNAME and volume serial number, and uses the SLDS for GENJCL, instead of the null RLDS.

DDNOUT1 is a required parameter on a COPY control statement. You can specify as many RECORD parameters as needed in a COPY control statement. If no RECORD parameter is specified, all log records are copied to the specified data set.

On a given COPY statement, the RECORD parameter and the DBRECOV parameter are mutually exclusive. You can specify multiple COPY control statements, but only one COPY statement with the DBRECOV parameter is allowed.

Two COPY statements must not specify the same output data set.

EXIT statement

An EXIT statement specifies that a user exit routine is to be used.

The format of the EXIT statement is:

NAME*(nnnnnnnn)*
Specifies the member name of the user exit. The user exit routine is accessed with a LOAD from the archive utility program; preferably binded into either JOBLIB or STEPLIB.

You can specify multiple EXIT control statements or multiple NAME parameters.

Return codes

The Log Archive utility provides the following return codes:

Code
Meaning

0
Archive processing completed successfully.

4
This return code is issued if one or both of the following events occur:

- Archive processing completed successfully, but not all OLDS were archived. A recovery point was encountered and end of job was forced. Rerun the Log Archive utility for the remaining unarchived OLDS. See SYSPRINT messages.
- An OLDS specified as input to the archive utility was already archived when this job ran. The SYSPRINT messages identify the OLDS that were already archived.

8
Archive processing completed unsuccessfully. Messages DFS3263I or DFS3062I indicate the reason.

U3274
ABEND—DBRC internal failure. Message DFS3274I plus various DSPxxxxx messages indicate the reason.

Examples of the DFSUARC0 utility

These examples show how to use the DFSUARC0 utility using the COPY control statement to create an RLDS and FEOV to ensure consistency in the SLDS.

Example 1

The following example shows the JCL for the Log Archive utility using the COPY control statement to create an RLDS:

```
//ARCHIVE  JOB MSGCLASS=A,CLASS=A,MSGLEVEL=(1,1)
//*
//ARC1    EXEC PGM=DFSUARC0,PARM='SYSA'
//STEPLIB   DD DSN=IMS.&SYS2..SDFSRESL,DISP=SHR
/*   COPY FROM 3 OLDS TO A SLDS                          */
/*   RLDS AND A USER DATA SET ARE ALSO CREATED           */
//DFSOLP00  DD DSN=OLP900,DISP=SHR,DCB=(BUFNO=20)
//DFSOLP01  DD DSN=OLP901,DISP=SHR,DCB=(BUFNO=20)
//DFSOLP02  DD DSN=OLP902,DISP=SHR
//DFSSLOGP  DD DSN=SLDSP.D82001.N001,DISP=(,KEEP),
//             UNIT=TAPE,VOL=(,,,99),LABEL=(,SL)
//RLDSDD1   DD DSN=RLDSP.D82001.N001,DISP=(,KEEP),
//             UNIT=TAPE,VOL=(,,,99),LABEL=(,SL)
//USERDD1   DD DSN=USER.D82001.N001,DISP=(,KEEP),
//             UNIT=3350,VOL=USER01,SPACE=(CYL,5)
//RECON1    DD DSN=RECON1,DISP=SHR
//RECON2    DD DSN=RECON2,DISP=SHR
//RECON3    DD DSN=RECON3,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//SYSUDUMP  DD SYSOUT=A
//SYSIN     DD *
 COPY DDNOUT1 (RLDSDD1) DBRECOV
 /*  THIS USER DATA SET CONTAINS                         */
 /*  X'A5', X'A6', AND X'A7' LOG RECORDS                 */
 COPY DDNOUT1 (USERDD1) -
  RECORD (O(5) T(X) V(A5) L(1) C(E)) -
  RECORD (O(5) T(X) V(A6) L(1) C(E)) -
  RECORD (O(5) T(X) V(A7) L(1) C(E))
 EXIT NAME (UEXIT01)
```

Example 2

The following example shows the JCL for the Log Archive utility using FEOV to ensure consistency in the SLDS.

```
//ARCHIVE2 JOB MSGCLASS=A,CLASS=A,MSGLEVEL=(1,1)
//*
//ARC2    EXEC PGM=DFSUARC0,PARM='SYSA'
//STEPLIB   DD DSN=IMS.&SYS2..SDFSRESL,DISP=SHR
//*   COPY FROM 2 OLDS TO DUAL SLDS                   */
//DFSOLP02  DD DSN=OLP902,DISP=SHR
//DFSOLP00  DD DSN=OLP900,DISP=SHR
//DFSOLS00  DD DSN=OLS900,DISP=SHR
//DFSOLS02  DD DSN=OLS902,DISP=SHR
//DFSSLOGP  DD DSN=SLDSP.D82001.N001,DISP=(,KEEP),
//             UNIT=TAPE,VOL=(,,,99),LABEL=(,SL)
//DFSSLOGS  DD DSN=SLDSS.D82001.N001,DISP=(,KEEP),
//             UNIT=TAPE,VOL=(,,,99),LABEL=(,SL)
//RECON1   DD  DSN=RECON1,DISP=SHR
//RECON2   DD  DSN=RECON2,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//SYSUDUMP DD  SYSOUT=A
//SYSIN    DD  *
  SLDS FEOV (08000)
 /*  THE SLDS ARE FORCED EOV AFTER 8000 LOG BLOCKS    */
 /*  ARE WRITTEN.                                     */
/*
```

Chapter 22. Log Merge utility (DFSLTMG0)

Use the Log Merge utility (DFSLTMG0) to produce a data set by merging the system log data sets (SLDS) from two or more IMS systems. The resulting data set is used as input to the Log Transaction Analysis utility.

The Log Merge utility identifies log records based on a system clock value in the record, then merges them in ascending order.

The Log Merge utility can merge up to nine IMS system logs. Each log is the output of a uniquely identified IMS system running during the same time span. The order of input to the Log Merge utility is LOG01, LOG02, LOG03, ..., LOG09.

DFSLTMG0 is placed in IMS.SDFSRESL during IMS system definition.

Subsections:

- "Restrictions" on page 503
- "Prerequisites" on page 503
- "Requirements" on page 503
- "Recommendations" on page 503
- "Input and output" on page 503
- "JCL specifications" on page 505

Restrictions

The Log Merge utility cannot use Common Queue Server (CQS) logs as input because CQS log records have a different format from IMS log records.

Prerequisites

Currently, no prerequisites are documented for the DFSLTMG0 utility.

Requirements

Currently, no requirements are documented for the DFSLTMG0 utility.

Recommendations

Currently, no recommendations are documented for the DFSLTMG0 utility.

Input and output

The input to the Log Merge utility consists of logs from up to nine separate IMS systems and control statements. A log from any single system can consist of a series of logs concatenated in time sequence. Log records must be from IMS systems that are running on processors with a synchronized external or internal clock to ensure that compatible system clock values between log records are produced. The system clock value, called the time of day (TOD) clock, is an 8–byte field stored at the end of each log record.

DFSLTMG0 produces as output a merged data set of log records made between the times specified with START and STOP control statements. This time is the Universal Time Coordinated (UTC).

© Copyright IBM Corp. 1974, 2024

Restriction: Do not use merged output as input to the Database Recovery utility.

Controlling the log merge

To control the log output:

- Choose the required systems that take part in the logical link paths you are examining.
- Choose logs from the required systems you want to examine when using the Log Transaction Analysis utility.
- Coordinate the series of input logs for each system so they cover a similar time span.
- Specify a start time and stop time for Log Merge utility control statements if you need to sample the cross-system processing for a particular time interval.

  You can give both start date (Julian) and time of day, or just time of day. These times apply to the first system log specified by the LOG01 DD statement. Other log activity is collected if it falls between the initial and final events present on the first log.
- Specify the control statement with the keyword listed under Log Record Selection to merge only certain types of log records.
- Specify MSG to select log records that are suitable for the transaction analysis step. (ALL records is the default, but this includes the DL/I activity for several systems in the utility input and this can cause extended processing time.)

Control statement format

START

Used to specify a start time. This statement must be present.

Table 27. START statement format

Position	Length	Value
1	5	START
6	1	blank
7	Variable	yyddd,hhmmsstt[{+|-}HHMM] where any trailing digits of hhmmsstt can be omitted and the optional time-zone information following hhmmsstt contains: **+ or -** Specifies the sign of the time-zone offset from UTC (Universal Coordinated Time). **HH** Specifies the number of whole hours of offset from UTC. HH can be a numeric value from 0 to 14. **MM** Specifies minutes of offset. MM can be 00, 15, 30, 45, or blank. You only need to specify the optional time-zone information if the offset to UTC on the day entered is different from the current offset, for example due to a daylight saving time change.

STOP

You must specify a stop time, which must be relative to the time field in LOG01.

Table 28. STOP statement format		
Position	**Length**	**Value**
1	4	STOP
5	1	blank
6	Variable	yyddd,hhmmsstt[{+|-}HHMM] where any trailing digits of hhmmsstt can be omitted and the optional time-zone information following hhmmsstt contains: **+ or -** Specifies the sign of the time-zone offset from UTC. **HH** Specifies the number of whole hours of offset from UTC. HH can be a numeric value from 0 to 14. **MM** Specifies minutes of offset. MM can be 00, 15, 30, 45, or blank. You only need to specify the optional time-zone information if the offset to UTC on the day entered is different from the current offset, for example due to a daylight saving time change.

Log Record Selection

Use this control statement to merge only certain types of log records. The format is free-form, starting in column 1. Any of the keywords in the following list can be used, in any combination desired, with the following syntax restrictions:

- BLANK, following a keyword terminates processing of this control statement.
- COMMA, following a keyword continues processing of this control statement.

Keyword
Meaning

ALL
All log record types are selected (this is the default if no control statements are present).

MSG
Selects all log records necessary for the Log Transaction Analysis utility (DFSILTA0); X'01', X'03', X'06', X'07', X'08', X'3x' series, X'40', X'42', X'47', X'48'.

3X
Selects all log records within the range; X'30' to X'3F'.

XX
Where XX is the log record type selected.

JCL specifications

EXEC statement

Executes the Log Merge utility DFSLTMG0.

```
//STEP0 EXEC PGM=DFSLTMG0
```

DD statements

STEPLIB DD

Points to IMS.SDFSRESL, which contains the IMS nucleus and required action modules.

```
//STEPLIB DD DSNAME=IMS.SDFSRESL,DISP=SHR
```

PRINT DD

Indicates the SYSPRINT data set used for control statements and error messages.

```
//PRINT DD SYSOUT=A
```

LOG01 DD

Describes the first input log data set.

```
//LOG01 DD DSNAME=IMS.LOGA,DISP=OLD,
//         VOL=SER=XXXXXX,UNIT=TAPE
```

LOG02 DD

Describes the second input log data set.

```
//LOG02 DD DSN=IMS.LOGB,DISP=OLD,
//         VOL=SER=XXXXXX,UNIT=TAPE
```

LOGOUT DD

Describes the output data set.

```
//LOGOUT DD DSN=IMS.LOGOUT,DISP=(,PASS),
//          VOL=SER=YYYYYY,UNIT=TAPE,
//          DCB=(RECFM=VBS,LRECL=6000,BLKSIZE=6008)
```

SYSIN DD

Describes the control statement data set.

```
//SYSIN DD *
START 75332,0830
STOP 75332,1030
MSG
```

Chapter 23. Log Recovery utility (DFSULTR0)

Use the Log Recovery utility (DFSULTR0) to produce a usable log data set from a log data set that contains read errors or that was not properly terminated.

The Log Recovery utility can recover both OLDSs and batch or online SLDSs.

This utility has four modes of operation:

CLS
Closes an OLDS from the write-ahead data set (WADS) or from the next OLDS.

CLS mode processes only OLDSs. To close SLDSs, use DUP mode. In CLS mode, a user-written Logger user exit (LOGWRT) is invoked during the execution of the Log Recovery utility if the exit routine is present. The Logger user exit (LOGWRT) is called once with an initialization call, once with a write call for each log buffer of data that is written, and once with a termination call.

Restriction: The Log Recovery utility does not support calling multiple LOGWRT user exits.

DUP
Processes either SLDSs or OLDSs. DUP mode creates an interim log containing error ID records, or a closed batch SLDS containing an end-of-file mark.

To safely close an SLDS, run DUP mode, then REP mode. Or, run DUP mode with a non-zero ERRC and the log sequence number (LSN) returned when the original error occurred. The system might issue message DFS616I, which includes the LSN, at the point of failure. If DFS616I is not issued, you must run DUP mode followed by REP mode to safely close an SLDS.

Attention: Do not run DUP mode without an LSN to close SLDSs in a production environment, unless you also run REP mode. Using DUP mode without also using an LSN or REP mode can result in loss of data.

REP
Reads the interim log, replaces the error ID records with user-specified data, and creates a new log.

PSB
Permits the generation of an "active PSBs" report from a mix of OLDS and SLDS.

The valid data set attributes for the input log data set are:

- RECFM=VB
- BLKSIZE greater than 8
- LRECL greater than 4 and less than or equal to BLKSIZE minus 4

The Log Recovery utility detects the following types of errors:

- I/O errors while reading the input log data set
- Errors in the log record or log block length
- Sequence errors in the log record, the log block, or the OLDS write time stamp for OLDS recovery only
- Log record sequence errors for SLDS recovery only

Subsections:

- “Restrictions” on page 508
- “Prerequisites” on page 508
- “Requirements” on page 508
- “Recommendations” on page 508
- “Output” on page 510
- “JCL specifications” on page 514
- “Utility control statements” on page 516

© Copyright IBM Corp. 1974, 2024

- “Return codes” on page 519

Restrictions

The following restriction apply to the DFSULTR0 utility.

- If single logging is used and DBRC is active, only single logs can be presented as input to the Log Recovery utility and only single logs can be created as output from DUP and REP mode. Otherwise, DBRC abends result.
- If dual logging is used and DBRC is active, only dual logs can be presented as input to the Log Recovery utility (except for PSB mode, which only accepts single log input). Otherwise, incorrect DBRC RECON updates result. If dual logs are presented as input, dual logs must be created as output from DUP and REP mode. You must correctly specify primary and secondary DSNAMEs on the DD statements.

Prerequisites

Before running the DFSULTR0 utility you must make sure that certain prerequisite tasks or conditions are met.

You must close OLDS and SLDS before they can be used as input to any utility.

OLDS recovery

An OLDS must be closed before it can be archived or used as input to any utility. The OLDS in use is closed automatically during normal shutdown or during an emergency restart. It must be closed using the Log Recovery utility if an emergency restart cannot close it, or when the OLDS is not closed because a write error is detected.

A stop time of zeros in the RECON indicates that the Log Recovery utility needs to be run in CLS mode. It should be run before DUP if possible; however, it can be run after REP.

SLDS recovery

An SLDS must be closed before it can be used as input to any utilities or IMS restart. The Log Recovery utility closes an SLDS created by a batch IMS system.

Requirements

Currently, no requirements are documented for the DFSULTR0 utility.

Recommendations

Currently, no recommendations are documented for the DFSULTR0 utility.

Input

The Log Recovery utility uses both single and dual logs for input. The utility accepts only input log data sets that are created by the same release of IMS as the utility release level.

Single log input

In CLS mode, the utility:

1. Reads the input log.
2. Produces a usable log if no errors are encountered.
3. Produces a report of active PSBs when the WADS is used as input. Because only the input log is used to generate the report, it may not be complete. PSB mode can be used to produce a complete report.

In DUP mode, the utility:

1. Reads the input log.
2. Creates a usable log if no errors are encountered.
3. Creates an interim log and an error listing if errors are encountered.

Using the interim log produced by DUP mode, and in REP mode, the utility:

1. Reads the interim log.
2. Copies good blocks to the output log.
3. Replaces error blocks with good ones based on user-specified control statements.
4. Produces a usable log.

In PSB mode, the utility:

1. Reads the input log.
2. Produces a report of active PSBs.

Dual log input

In the following discussion, the terms "primary" and "secondary" are used to identify the two logs of a dual log data set.

In CLS mode, the utility:

1. Reads the input logs.
2. Produces a usable log if no errors are encountered at the same point on both OLDS. If an error is encountered on one OLDS but not the other, an error listing with an error block ID of NONE is produced and the utility continues processing. In this case, the OLDS pair produced may be usable as input to an IMS restart or archive (which also tolerate errors on only one of a pair of OLDS), but DUP mode processing is needed to remove the errors.
3. Produces a report of active PSBs when the WADS is used as input. Because only the input log is used to generate the report, it may not be complete. PSB mode can be used to produce a complete report.

In DUP mode, the utility:

1. Reads the primary log and copies the contents to a new system log. If it encounters an error block, DUP mode positions a read operation on the secondary log where the log error was encountered. DUP mode then reads the secondary log and copies the contents to the same new system log. If an error is now encountered on the secondary log (but not at the same position), DUP mode positions a read operation on the primary log where the error was encountered. This process continues until a complete new system log is produced. The following figure illustrates DUP mode and REP mode using dual logging.

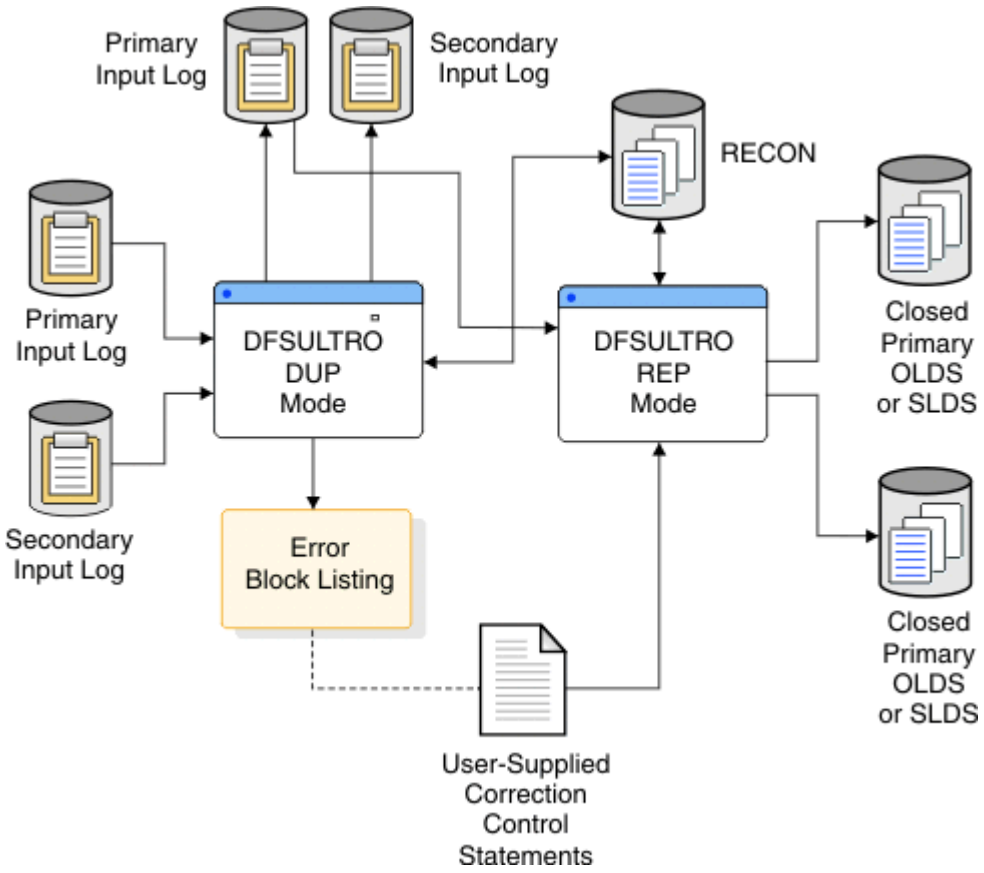


Figure 36. DUP mode and REP mode when dual logging is used

2. Copies both error blocks onto the interim log and uniquely identifies the error blocks when it encounters an error on both logs in the same position. The interim log data set contains all valid log blocks, error blocks, and error ID records.
3. Produces a character and hexadecimal listing of the error blocks to be used as a guide for creating the user-specified control statements required by REP mode.

Using dual logs for input, REP mode:

1. Reads the interim log created by DUP mode
2. Copies good blocks
3. Replaces error blocks with good ones based on control statements
4. Produces a usable log

If dual system log input is used and errors at the same position on both input logs are not encountered, the log produced by DUP mode is correct and REP mode is not required.

Output

In addition to the usable log, active PSB report, and the interim log, the Log Recovery utility also produces the following:

- Interim Log Error ID Record
- Error Block Listing (SYSPRINT)
- REP mode verification messages
- Dump of data record

Interim log error ID record

The following figure illustrates the error ID record on the interim log produced from dual log input. In this example, BLK2 of both the primary and secondary logs has errors. On the interim log, the first error ID is for BLK2B and the second error ID is for BLK2A. During REP mode, BLK2A or BLK2B is replaced with a good block based on control statements. This example also shows the valid log after REP mode execution.

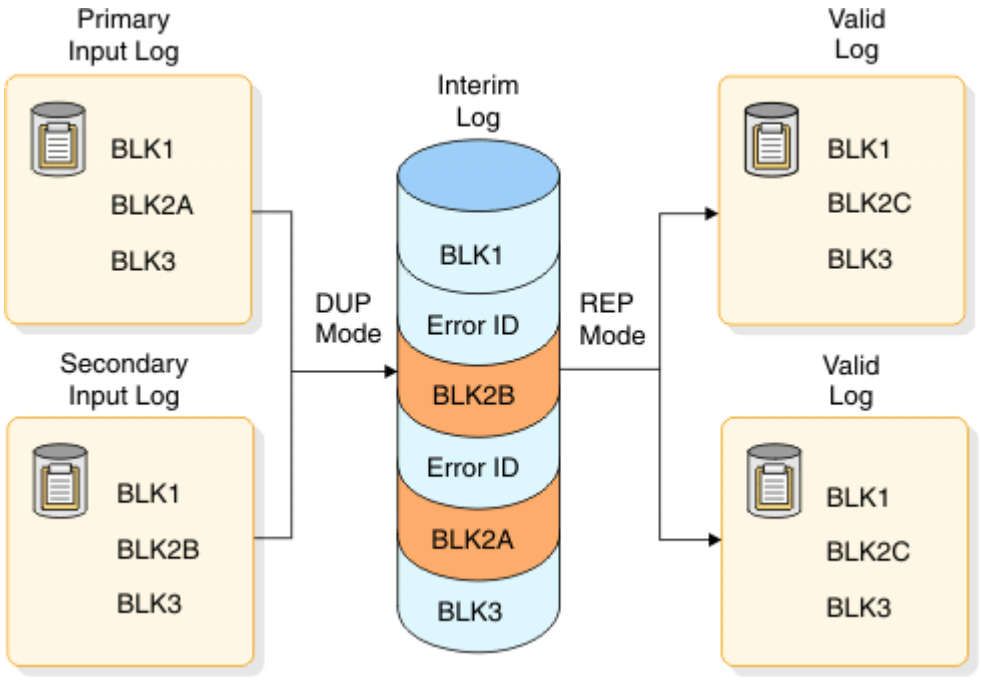


Figure 37. Error ID records on an interim log

Error block listing (SYSPRINT)

The error block listing contains the errors found during execution of CLS mode and DUP mode. It also contains verification messages resulting from REP mode followed by a dump of the data record.

The following error listing is for both CLS mode and DUP mode.

```
ppppppppppppppppppp ON dddddddd BLOCK# bbbbbbb ** ERROR-ID=xnnnnn **
sssssssssssssssssss--gghhiijj
```

The fields of the error block listing are:

ppppppppppppppppppp
Is a message prefix which identifies the type of error. The following types of errors are identified:

PERMANENT I/O ERROR
The SYNAD exit for the input log was entered with an error other than a data check or a length error or consecutive data checks occurred.

DATA CHECK
The SYNAD exit for the input log was entered with a data check error.

END-OF-DATA
The EODAD exit for the input log was entered. This is not an error but rather an indication that processing for this input data set has ended. If the swap to the alternate log is successful, processing will continue on the alternate log.

BLOCK LENGTH ERROR
The length in the block descriptor word (BDW) is not valid.

BLOCK TOD ERROR
The time-of-day (TOD) in the OLDS block suffix is not in ascending order.

BLOCK SEQ ERROR
The block sequence number in the OLDS block suffix is not in ascending order.

RECORD LENGTH ERROR
The length in a record RDW is not valid.

RECORD SEQ ERROR
The record sequence number is not in ascending order.

ddddddd
Is the ddname of the data set where the error is encountered. The following list shows possible ddnames:

IEFRDER
The primary input SLDS.

IEFRDER2
The secondary input SLDS.

DFSOLP
The primary input OLDS.

DFSOLS
The secondary input OLDS.

bbbbbbb
Is the relative block number (in hexadecimal) of the block in error. Blocks are counted beginning with the first block of the first input volume, starting with 0000001.

x
Is either an A or a B and identifies whether the error occurred on the current log or the alternate log. When processing begins, the primary log is the current log and the secondary log is the alternate log. If processing swaps to the alternate because of an error, these roles reverse and processing continues. Errors on the alternate log are always reported before errors on the current log.

nnnnn
Is a sequential number which identifies the error.

xnnnnn is 'NONE ' when CLS mode processing on dual OLDS encounters an error on one OLDS but not on the other at some point. The reason for the error listing under these conditions is to alert you to a situation where you might want to use DUP mode to fix the errors even though the OLDS may be usable for restart or archive without doing so.

sssssssssssssssssss
Is a message suffix which further identifies the error. This suffix can be:

ORIGINAL BDW X'ssss'
The original block length in the BDW is not correct and has been changed. The variable *ssss* is the original value expressed in hexadecimal notation.

RCD AT OFFST X'oooo'
A log record has an invalid length in the record descriptor word (RDW). The variable *oooo* is the offset (relative to zero), in hexadecimal, from the beginning of the block to the RDW in error.

ffffffff* TO *tttttttt
A block sequence, block TOD, or record sequence error has occurred. The variable *ffffffff* is the last good value (or assumed good value). The variable *tttttttt* is the value in error. After a sequence error occurs, the block sequence number, the block TOD, and the first record sequence number in the next block are assumed to be good, and thus begin a new sequence on which the remaining records will be checked. The Log Recovery utility reports breaks in sequences of good data. You must analyze the reports and determine what is valid data and what is invalid data.

gg
Is either blank or NS. This is the first of several special suffix values. NS applies only with dual SLDS input. The two input logs do not start with the same block. It is not possible to swap to the 'alternate'

log (or write to the corresponding output data set) until the first block common to both input logs is read.

hh
Is either blank or CE. This is the second of the special suffix values. CE indicates that this is a consecutive error. A second through nth error has occurred without reading an intervening good block.

ii
Is either blank or SA. This is the third of the special suffix values. SA indicates that it is not possible to swap to the alternate log because the alternate log has already either reached END-OF-DATA or encountered a PERMANENT I/O ERROR.

jj
Is either blank or SO. This is the last of the special suffix values. SO indicates that during a swap to the alternate log has either reached END-OF-DATA or encountered a PERMANENT I/O ERROR. In this case processing would normally return to the original current log. However, the current log has already reached END-OF-DATA or encountered a PERMANENT I/O ERROR. Therefore it is not possible to return to the current log.

REP mode verification messages

During REP mode processing, a valid replacement of data on the interim log data set causes the following message to be printed:

```
DATA REPLACED IN RECORD Axxxxx ... replacement data text...
```

Where xxxxx is the error ID.

An error in the control statement format causes the following message to be printed:

```
ERROR IN CONTROL STATEMENT FORMAT ... text of control statement...
```

Dump of data record

The dump of the data record following the verification messages is a hexadecimal representation of the record. The hexadecimal representation is printed in four lines per print line of the data record.

- The first line consists of the position within the block in error (starting with 1), and the EBCDIC representation of the bytes.
- The second line indicates the first byte of each log record, using an asterisk.
- The third line consists of the zone half representation.
- The fourth line consists of the digit half representation.

The following figure shows the format of the printed output.

```
000001          q    RRE  b        // EBCDIC representation
          *                        // first byte of a log record
       2000020049  00DDC40809      // high-order hexadecimal digit
       00000D0008  029954024F      // low-order hexadecimal digit
```

Active region messages

When WADS is specified in CLS mode, the active PSBs at the time of the system failure are printed. A line is printed for each PSB active at the time of failure. If backout is required for the PSB, database names are listed under the PSB line in the output. The following figure shows the format of this output.

```
****************** RECOVERY REQUIREMENTS **********************

PSB NAME   RECOVERY TOKEN              DATABASE   DSID   ACTION Required
PPPPPPPPP
```

```
            EEEEEEEEHHHHHHHHHHHHHHHH

                                          DDDDDDDD
NNN     MMMMMMMM
                                          SSSSSSSS


END OF REPORT
```

The fields in the report have the following meanings:

PPPPPPPP
The PSB name.

EEEEEEEE
The EBCDIC portion of the recovery token.

HHHHHHHHHHHHHHHH
The hexadecimal portion of the recovery token for eight bytes (16 characters).

DDDDDDDD
The database name.

SSSSSSSS
The database name status. If no database names are in the DDDDDDDD field, one of the following messages appears:

No database names found
DBNAME list may be incomplete

NNN
The Fast Path data set ID number that indicates the area data set.

MMMMMMMM
The message issued. One of the following messages is issued:

Backout is required
Redo is required
Databases are in doubt

The Active-Region report is also produced in PSB mode.

JCL specifications

The following JCL is required to run DFSULTR0.

EXEC statement

Invokes the Log Recovery utility (DFSULTR0). The format must be:

```
//STEP EXEC PGM=DFSULTR0,PARM='IMSID=iiiiiiii,
//             DBRC=ddd, IMSPLEX=imsplex_name, DBRCGRP=xxx'
```

IMSID=*iiiiiiii*
Indicates the IMSID of the on-line system that created the input OLDS.

Requirement: IMSID= is required for CLS mode. IMSID= is required for DUP mode with OLDS input and DBRC=YES (specified or defaulted).

IMSID= is ignored if it is specified but not needed.

DBRC=YES|NO
Indicates that the DBRC= default is not established by the IMSCTRL macro during IMS system definition.

DBRC=NO (or N) can be specified to explicitly declare that DBRC is not to be used for this execution of this utility.

DBRC=YES (or Y) can be specified to explicitly declare that DBRC is to be used for this execution of this utility. DBRC=YES is required (and the default) for CLS mode. DBRC=YES is optional for DUP and REP modes.

Recommendation: If DUP mode is run with DBRC active, REP mode should also be run with DBRC active.

IMSPLEX=*imsplex_name*
Indicates which IMSplex DBRC should join. IMSPLEX= is an optional parameter.

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

To allow a parameter to default, the complete parameter (including the keyword) must be omitted from the PARM field.

If no input parameters are specified, the default will be IMSID=(not specified) and DBRC=YES.

DD statements

The DD statements are only used if they are required for a given execution of the Log Recovery utility.

Restrictions:

- Specify OLDS input using the DFSOLP (and DFSOLS) DD statement.
- Specify SLDS input using the IEFRDER (and IEFRDER2) DD statement.
- Do not specify DFSOLP (and DFSOLS) DD statements in an execution that also contains an IEFRDER (and IEFRDER2) DD statement.
- Do not specify DFSWADSn DD statements in an execution that also contains a DFSNOLP (and DFSNOLS) DD statement.
- Do not specify DFSWADSn, DFSNOLP (and DFSNOLS), or any combination in an execution that also contains an IEFRDER (and IEFRDER2) DD statement.
- Do not specify DFSPOLP (and DFSPOLS) DD statements in an execution that also contains a DFSNOLP (and DFSNOLS) DD statement.
- Do not specify DFSWADSn DD statements in an execution that also contains the keyword NOWADS.

STEPLIB DD
Points to IMS.SDFSRESL, which contains the Log Recovery utility's modules.

SYSPRINT DD
Defines the system messages data set.

SYSUDUMP DD
Defines the dump data set.

SYSUDUMP statements are not included in the examples at the end of this utility.

DFSOLP DD
Defines the primary, or only, input OLDS.

DFSOLS DD
Defines the secondary input OLDS. Include this statement only when dual OLDSs are used.

DFSWADS*n* DD
Defines the WADS data set, where *n* can be 0 through 9. All WADSs used during online execution can be specified, but only those in use by the online system at the time of failure are required. This DD statement is required when closing an OLDS from a WADS. If no WADS were in use by the online system, no DFSWADSn DD statements are used.

DFSNOLP DD
Defines the primary, or only, next-OLDS. The next-OLDS is the OLDS written by the online IMS system immediately after the OLDS having a write error.

DFSNOLS DD
Defines the secondary next-OLDS. Include this statement only when dual OLDSs are used.

DFSPOLP DD
Defines the primary OLDS that the IMS online subsystem used before the specified OLDS which is being closed. If there is no prior OLDS, this DD statement should not be used.

DFSPOLS DD
Defines the secondary OLDS that the IMS online subsystem used before the specified OLDS that is being closed. Include this statement only when dual prior OLDS are used.

IEFRDER DD
Defines the primary, or only, input SLDS. All input SLDS logs for DUP mode should have the same block size. IEFRDER is used to specify the concatenation of OLDS and SLDS logs for PSB mode. When specifying a concatenation of logs, the names of the logs must be provided in ascending order.

IEFRDER2 DD
Defines the secondary input SLDS. Include this statement only when dual logs are used. Omit this statement if you do not need the data sets. Do not use DD DUMMY or DSNAME=NULLFILE.

NEWRDER DD
Defines the primary, or only, output data set for the new or interim log.

NEWRDER2 DD
Defines the secondary output data set for the new or interim log. If DBRC is active and dual logs are used as input, this statement is required. If DBRC is not active, this statement is not required. Do not use DD DUMMY or DSNAME=NULLFILE.

RECON1 DD
Defines the first DBRC RECON data set. This statement is not required if dynamic allocation is used.

RECON2 DD
Defines the second DBRC RECON data set. This statement is not required if dynamic allocation is used.

RECON3 DD
Defines the optional DBRC RECON data set used when an error is encountered in RECON1 or RECON2. This RECON data set must be the same RECON data set used by the control region. This statement is not required if dynamic allocation is used.

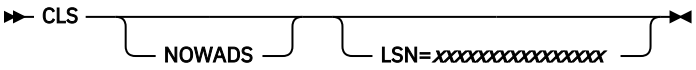
SYSIN DD
Defines the control data set containing the log recovery input control statements.

Utility control statements

Utility control statements for the Log Recovery utility differ depending on whether the CLS mode, DUP mode, REP mode, or PSB mode is used.

CLS mode–Close an OLDS from the WADS or next OLDS

The format of this control statement is:



CLS
Indicates CLS mode.

Requirement: DBRC is required for CLS mode.

When closing from the WADS, if a prior OLDS is available, the suffix from the last block written to the prior OLDS (the block sequence number is passed to DBRC at OLDS switch and stored in the RECON) is obtained. The block suffix is used to establish a basis for sequence checking the OLDS being closed.

When closing from the WADS, either EOF or encountering the first error causes an attempt to close the OLDS from the WADS. If a sequence error is found, CLS mode fails. A listing containing the block at the first error is produced.

When closing from the next-OLDS, the sequence number of the first block of the next-OLDS (BSN) is determined. The input OLDS is closed when block BSN-1 is found on the input OLDS. If either EOF or an error is encountered before block BSN-1 is found, CLS mode fails.

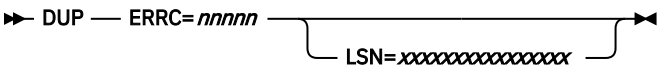
NOWADS
Suppresses the use of WADS when closing the OLDS. When this keyword is used, the DFSWADSn DD card must be removed from the JCL. Otherwise, user abend U3271 will result.

Attention: Use NOWADS only when WADS is unavailable. Do not use this keyword if possible; log records can be lost, data integrity can be compromised, and recovery might not be complete.

LSN=*xxxxxxxxxxxxxxxx*
An optional parameter used in DUP or CLS mode processing to specify a log sequence number that must be encountered on the input log. If the utility would otherwise succeed (return code of 0 or 4) but the last log sequence number encountered is less than *xxxxxxxx*, the utility ends with a return code of 8, DBRC is not notified of a successful completion, and message DFS3271I is issued. The value of *xxxxxxxxxxxxxxxx* must be 16 hexadecimal characters.

DUP mode–Recover an OLDS or SLDS (create an interim log)

The format of this control statement is:



DUP
Indicates DUP mode.

ERRC=*nnnnn*
Is used to terminate DUP mode after a predefined number of I/O or sequence errors are detected on the input log data set. *nnnnn* specifies the number of errors (00000 through 99999). If no value is specified or the keyword is omitted, the default is 99999. This field must contain 5 digits, with leading zeros.

If an *nnnnn* of 00000 is specified, DUP mode is terminated and the interim log is closed when the first error is encountered. The error ID record and error blocks are not written on the interim log. REP mode is not required.

ERRC=00000 is used to close an SLDS without having to run REP mode. A listing can be produced that contains the block at the first error. When the first error is encountered, additional checks are made to ensure that no newer data exists beyond the first error.

Attention: Use caution when running DUP ERRC=00000 in a production environment. Because these checks are not foolproof, only specify ERRC=00000 if you clearly understand the risks involved: closing the log in the middle of good data, for example, can destroy good data. It is safer to run with a value of *nnnnn* greater than 00000 and to also run REP mode.

If an ERRC value greater than zero is specified, DUP mode is terminated when either EOF is encountered or ERRC is reached (ERRC is tested before each block read). If errors are found, error ID records and error blocks are written on the interim log and REP mode is required. A listing that contains the errors found is produced.

Specify an ERRC value greater than zero when recovering an OLDS or SLDS.

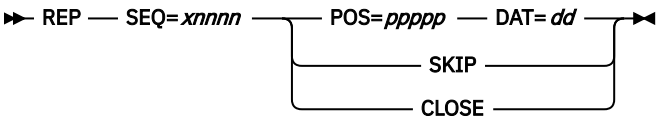
LSN=*xxxxxxxxxxxxxxxx*
An optional parameter used in DUP or CLS mode processing to specify a log sequence number that must be encountered on the input log. If the utility would otherwise succeed (return code of 0 or 4) but the last log sequence number encountered is less than *xxxxxxxx*, the utility ends with a return

code of 8, DBRC is not notified of a successful completion, and message DFS3271I is issued. The value of *xxxxxxxxxxxxxxxx* must be 16 hexadecimal characters.

REP mode–Recover an OLDS or SLDS (create a new log)

This mode reads the interim log created by DUP mode, copies good blocks, and replaces error blocks with good ones based on the REP control statements. (Only the primary input data set is read during REP mode). The output log data set is a new OLDS or SLDS log. At least one control statement is required but any number can be included.

The format of the control statement is:



REP
Indicates REP mode.

SEQ=*xnnnnn*
Indicates the identification number of the block to be changed. The number is provided in the DUP mode listing output.

POS=*pppppp*
Indicates the starting position, relative to 1, of the data being replaced.

DAT=*dd*
dd is 2 to 50 hexadecimal characters (0 through 9, A through F) representing the replacement data.

SKIP
Indicates the output log will not contain this block of data.

CLOSE
Indicates the output log will be closed immediately before this error block.

The REP mode CLOSE option should not be confused with the process of closing an OLDS from the WADS or next-OLDS using CLS mode.

The following rules apply to use of the REP statement:

- At least one control statement must be supplied.
- Unless the log is closed at a prior block, each error block identified in the DUP mode output must have at least one control statement supplied for it.
- When multiple REP statements are provided, the identification numbers (SEQ=) must be in ascending block number sequence.
- If a block is identified as being in error even though the data is good, a control statement must be supplied for the block. Replace the first 4 bytes of the good block with the existing data. This is usually the case for the first block following an I/O error.
- If dual logs are used in DUP mode, supply a statement for only one of the two blocks in error, either Annnnn or Bnnnnn. The block not selected is ignored and is not written to the output log.
- If the log being recovered is an OLDS which has not been properly closed from either the WADS or next OLDS, the Log Recovery utility must be rerun in CLS mode using the output of REP mode as input.

PSB mode—Print report of active PSBs

PSB mode is used when a previous execution of this utility issued the following message:

```
DFS3272I X'47' LOG RECORD NOT FOUND.
ACTIVE PSB MESSAGES NOT GENERATED.
```

To get active region messages, the Log Recovery utility must be rerun in PSB mode. PSB mode can be used at any time to determine which PSBs are active.

PSB mode should not be run with an OLDS that is open. An incomplete listing results.

The format of this control statement is:

PSB
Indicates PSB mode.

Return codes

The Log Recovery utility provides the following return codes:

Code
Meaning

0
Successful completion. If running CLS mode to terminate OLDS with WADS, ignore any error messages.

4
The report may be incomplete. This occurs when Log Recovery does not find certain log records.

If Log Recovery did not find a complete set of 47 records, message DFS3272I is issued, the report output will contain ACTIVE PSB MESSAGES NOT GENERATED, and no other report output is generated. This is shown in the following message:

```
DFS3272I X'47' LOG RECORD NOT FOUND.
ACTIVE PSB MESSAGES NOT GENERATED.
```

If Log Recovery did not find a 5607 record for a unit of recovery of the PSB, the DB status for that unit of recovery contains DBNAME LIST MAY BE INCOMPLETE.

In either case, you must include earlier log data so that the needed records are included. If you were running CLS mode, you must run PSB mode to include any logs in addition to the last OLDS.

8
Unsuccessful completion. If the problem is due to a mismatch of the log release level and the utility release level, message DFS3062I also accompanies this error code.

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

Related reference
LOGWRT: Logger user exit (DFSFLGX0 and other LOGWRT exits) (Exit Routines)

Examples of the DFSULTR0 utility

These examples show how to use the DFSULTR0 utility to recover OLDS or SLDS in various modes.

Example 1

The following example shows how to close an OLDS from the WADS using CLS mode. The input data set is closed in place. The DBRC RECON data set is updated with the "close time."

```
//EXAMPL01  JOB ........
//*
//DFSULTR0 EXEC PGM=DFSULTR0,PARM='IMSID=iiiiiiii'
//*
//* NOTE - IMSID= is required
//* NOTE - Defaults are DBRC=YES
//* NOTE - DBRC=NO is not valid.
//*
//SYSPRINT  DD SYSOUT=A
//DFSOLP    DD ........  Primary   OLDS to be closed
//DFSOLS    DD ........  Secondary OLDS to be closed
//DFSPOLP   DD ........  Primary prior OLDS
//DFSPOLS   DD ........  Secondary prior OLDS
//DFSWADSn  DD ........  WADS used by on-line system
//RECONn    DD ........  DBRC RECON data set(s)
//*                      (can be dynamically allocated)
//SYSIN     DD *
CLS
```

If no WADS were in use when the input OLDS or prior OLDS was created, remove the DFSWADSn DD statement and add the NOWADS keyword to the control statement.

If no prior OLDS are available, remove the DFSPOLP (and DFSPOLS) DD statement.

Example 2

The following example shows how to close an OLDS from the next-OLDS using CLS mode. The input data set is closed in place. The DBRC RECON data set is updated and the flag is turned off.

```
//EXAMPL02  JOB ........
//*
//DFSULTR0 EXEC PGM=DFSULTR0,PARM='IMSID=iiiiiiii'
//*
//* NOTE - IMSID= is required
//* NOTE - Defaults are DBRC=YES
//* NOTE - DBRC=NO is not valid
//*
//SYSPRINT  DD SYSOUT=A
//DFSOLP    DD ........  OLDS to be closed from next-OLDS
//DFSNOLP   DD ........  next-OLDS
//RECONn    DD ........  DBRC RECON data set(s)
//*                      (can be dynamically allocated)
//SYSIN     DD *
CLS
```

Example 3

The following example shows how to use DUP mode as the first of two steps in the recovery of an OLDS. The input data set is copied to an interim data set. Interim log records are created in the DBRC RECON.

```
//EXAMPL03  JOB ........
//*
//DFSULTR0 EXEC PGM=DFSULTR0,PARM='IMSID=iiiiiiii'
//*
//* NOTE - IMSID= is required
//* NOTE - Defaults are DBRC=YES
//*
//SYSPRINT  DD SYSOUT=A
//DFSOLP    DD ........  Primary   OLDS to be recovered
//DFSOLS    DD ........  Secondary OLDS to be recovered
//NEWRDER   DD ........  Primary   interim data set
//NEWRDER2  DD ........  Secondary interim data set
//RECONn    DD ........  DBRC RECON data set(s)
//*                      (can be dynamically allocated)
//SYSIN     DD *
DUP ERRC=nnnnn
```

If an ERRC value greater than zero is specified (the default is 99999), error blocks are written to the output data set, and a listing is produced for the blocks in error. REP mode is required to correct the

errors and to remove error blocks. If no errors are found and the execution is successful, REP mode is not required.

When ERRC=00000 is specified, NEWRDER (and NEWRDER2) is closed when EOF or the first error is encountered on DFSOLP (and DFSOLS). If the execution is successful, REP mode is not required. If the execution is unsuccessful, DUP mode should be rerun with an ERRC value greater than zero and REP mode is required.

If the log being recovered is an OLDS which has not been properly closed from either the WADS or next OLDS, the Log Recovery utility must be rerun in CLS mode using the output of REP mode as input (or the output of DUP mode if no errors were detected).

Example 4

The following example shows how to use REP mode as the second of two steps in the recovery of an OLDS. The input data set is copied to a new OLDS. During the copy process, error blocks are removed and the blocks in error are corrected as directed by the REP control statements. The interim data set information in the DBRC RECON is deleted. The original OLDS information in the DBRC RECON is replaced by the output data set information.

```
//EXAMPL05  JOB ........
//*
//DFSULTR0 EXEC PGM=DFSULTR0,PARM='IMSID=iiiiiiii'
//*
//* NOTE - IMSID= is required
//* NOTE - Defaults are DBRC=YES
//*
//SYSPRINT  DD SYSOUT=A
//DFSOLP    DD ........  Primary   interim data set
//DFSOLS    DD ........  Secondary interim data set
//NEWRDER   DD ........  Primary   recovered OLDS
//NEWRDER2  DD ........  Secondary recovered OLDS
//RECONn    DD ........  DBRC RECON data set(s)
//*                      (can be dynamically allocated)
//SYSIN     DD *
REP  SEQ=A00001   POS=000018  DAT=83 (EXAMPLE ONLY)
REP  SEQ=A00002   SKIP
REP  SEQ=A00003   CLOSE
```

If the log being recovered is an OLDS which has not been properly closed from either the WADS or next OLDS, the Log Recovery utility must be rerun in CLS mode using the output of REP mode as input.

Example 5

The following example shows how to use DUP mode as the first of two steps in the recovery of an SLDS. The input data set is copied to an interim data set. Interim log records are created in the DBRC RECON.

```
//EXAMPL04  JOB ........
//*
//DFSULTR0 EXEC PGM=DFSULTR0
//*        (PARM NOT REQUIRED - SEE NOTES BELOW)
//*
//* NOTE - IMSID= is ignored
//* NOTE - Defaults are DBRC=YES
//*
//SYSPRINT  DD SYSOUT=A
//IEFRDER   DD ........  Primary   SLDS to be recovered
//IEFRDER2  DD ........  Secondary SLDS to be recovered
//NEWRDER   DD ........  Primary   interim data set
//NEWRDER2  DD ........  Secondary interim data set
//RECONn    DD ........  DBRC RECON data set(s)
//*                      (can be dynamically allocated)
//SYSIN     DD *
DUP ERRC=nnnnn
```

If an ERRC value greater than zero is specified (the default is 99999), error blocks are written to the output data set and a listing is produced for the blocks in error. REP mode is required to correct the errors and to remove error blocks. If no errors are found and the execution is successful, REP mode is not required.

Example 6

The following example shows how to use REP mode as the second of two steps in the recovery of an SLDS. The input data set is copied to a new SLDS. During the copy process, error blocks are removed and the blocks in error are corrected as directed by the REP control statements. The interim data set information in the DBRC RECON is deleted. The original SLDS information in the DBRC RECON is replaced by the output data set information.

```
//EXAMPL06  JOB ........
//*
//DFSULTR0 EXEC PGM=DFSULTR0
//*        (PARM NOT REQUIRED - SEE NOTES BELOW)
//*
//* NOTE - IMSID= is ignored
//* NOTE - Defaults are DBRC=YES
//*
//SYSPRINT  DD SYSOUT=A
//IEFRDER   DD ........  Primary   interim data set
//IEFRDER2  DD ........  Secondary interim data set
//NEWRDER   DD ........  Primary   recovered SLDS
//NEWRDER2  DD ........  Secondary recovered SLDS
//RECONn    DD ........  DBRC RECON data set(s)
//*                      (can be dynamically allocated)
//SYSIN     DD *
REP  SEQ=A00001    POS=000018  DAT=83 (EXAMPLE ONLY)
REP  SEQ=A00002    SKIP
REP  SEQ=A00003    CLOSE
```

Example 7

The following example shows how to generate a listing of "active PSBs" after having received message DFS3272I X'47' LOG RECORD NOT FOUND. ACTIVE PSB MESSAGES NOT GENERATED. PSB mode is used.

```
//EXAMPL07  JOB ........
//*
//DFSULTR0 EXEC PGM=DFSULTR0
//*
//* NOTE - IMSID= is ignored
//* NOTE - DBRC=YES is not valid
//* NOTE - Defaults are DBRC=NO
//*
//SYSPRINT  DD SYSOUT=A
//*
//* NOTE - The first log data set in the IEFRDER DD statement should
be the latest log data set containing the X'47' record.
//*
//IEFRDER  DD ........ next or prior OLDS or SLDS
//         DD ........ next or prior OLDS or SLDS
//         DD ........ next or prior OLDS or SLDS
⋮
//         DD ........ latest OLDS or SLDS
//*
//*
//SYSIN     DD *
PSB
```

The input logs must be concatenated in the sequence in which they were created, and there must not be any overlap or gap in log record content.

Example 8

The following example shows how to generate a listing of "active PSBs" from a concatenation of input logs (OLDS and SLDS). PSB mode is used.

```
//EXAMPL08  JOB ........
//*
//DFSULTR0 EXEC PGM=DFSULTR0
//*
//* NOTE - IMSID= is ignored
//* NOTE - DBRC=YES is invalid
//* NOTE - Defaults are DBRC=NO
//*
//SYSPRINT  DD SYSOUT=A
```

```
//IEFRDR    DD ........  OLDS or SLDS
⋮
            DD ........  OLDS or SLDS
⋮
//          DD ........  OLDS or SLDS
//*                           (see note below)
//SYSIN     DD *
PSB
```

Requirement: The input logs must be concatenated in the sequence in which they were created. If OLDS and SLDS are mixed, there must not be any overlap in log record content.

Example 9

The following example shows how to close an SLDS created by IMS batch, using DUP mode and ERRC=00000. The input data set is copied to, and closed in, the output data set. The input SLDS information in the DBRC RECON is replaced by the output data set information.

```
//EXAMPL09  JOB ........
//*
//DFSULTR0 EXEC PGM=DFSULTR0
//*        (PARM NOT REQUIRED - SEE NOTES BELOW)
//*
//* NOTE - IMSID= is ignored
//* NOTE - Defaults are DBRC=YES
//*
//SYSPRINT  DD SYSOUT=A
//IEFRDER   DD ........  Primary   SLDS to be closed
//IEFRDER2  DD ........  Secondary SLDS to be closed
//NEWRDER   DD ........  Primary   output SLDS
//NEWRDER2  DD ........  Secondary output SLDS
//RECONn    DD ........  DBRC RECON data set(s)
//*                      (can be dynamically allocated)
//SYSIN     DD *
DUP ERRC=00000
```

When ERRC=00000 is specified, NEWRDER (and NEWRDER2) is closed when EOF or the first error is encountered on IEFRDER (and IEFRDER2). If the execution is successful, REP mode is not required. If the execution is unsuccessful, DUP mode should be rerun with an ERRC value greater than zero and REP mode is required.

If the input SLDS (IEFRDER, IEFRDER2) is a multiple volume tape data set, only the last volume needs to be specified on the DD statement. In addition, the data set name (DSN) on the output DD statement (NEWRDER, NEWRDER2) should be the same as the input. If the execution is successful, only the volume information is replaced in the DBRC RECON. ERRC=00000 is required.

Part 5. Service utilities

Use the service utilities to perform IMS maintenance and operational functions.

Each topic introduces how the utility works, defines requirements and restrictions for its use, and provides examples.

© Copyright IBM Corp. 1974, 2024

Chapter 24. Batch SPOC utility (CSLUSPOC)

Use the Batch SPOC (Single Point of Control) utility to submit IMS operator commands to members of the IMSplex.

In the Batch SPOC utility, the program parameters define the IMSplex environment and the input to the utility is IMS operator commands. Output from the utility is command responses written to the SYSPRINT file.

Subsections:

- "Restrictions" on page 527
- "Prerequisites" on page 527
- "Requirements" on page 527
- "Recommendations" on page 527
- "Input and output" on page 527
- "JCL specifications" on page 528
- "Return codes" on page 529

Restrictions

You must use commands that are supported by the OM API with the Batch SPOC utility. These commands are all type-2 commands and a subset of type-1 commands only. For more on the supported commands, see Commands and keywords supported by the OM API (Commands).

For SYSIN input with a fixed length record without a sequence number, do not use all numeric parameters in the last 8 bytes of the record because the parameter will be considered a sequenced number and will be truncated.

For SYSIN input with a variable length record, a sequence number is not allowed.

Prerequisites

Currently, no prerequisites are documented for the Batch SPOC utility.

Requirements

The Batch SPOC utility only works in an IMSplex environment.

Recommendations

Currently, no recommendations are documented for the Batch SPOC utility.

Input and output

The input for the Batch SPOC utility is the SYSIN DD statement. An example of a SYSIN DD statement is:

```
//SYSIN     DD *
 QRY IMSPLEX SHOW(MEMBER,TYPE,STATUS)
```

You provide the SYSIN file that contains the commands that you want to execute. The commands are executed serially. When one command completes, the next command is executed until all records from

© Copyright IBM Corp. 1974, 2024

the SYSIN file are processed. Continuation of the SYSIN control statements is specified by a plus sign ('+') or a minus sign ('-') as the last non-blank character of the line. A plus sign removes the leading spaces from the next line; a minus sign keeps leading spaces.

Comments can be included within the SYSIN file. Use the following structure for comments:

- On a comment line, the comment begins with an asterisk (*) in column 1 and no command is on the same line. For example:

  ```
  * this is a comment
  ```

  When a comment is on its own line, the Batch SPOC utility does not send the comment line to OM for processing.
- On the same line as the command, the comment starts with a "/*" and ends with a "*/". For example:

  ```
  /* this is a comment */
  ```

  A comment in the following position is supported:

  ```
  QRY IMSPLEX SHOW(ALL)   /* This is a comment */
  ```

  Comments in the following positions are not supported:

  – A comment is in front of a command on the same line. For example:

    ```
    /* This is a comment */  QRY IMSPLEX SHOW(ALL)
    ```

  – A comment is in the middle of a command on the same line. For example:

    ```
    QRY  /* This is a comment */  IMSPLEX  SHOW(ALL)
    ```

  – Comments span over multiple lines in a block. For example:

    ```
    /* This is comment line number 1,
    this is comment line number 2    */
    ```

  When a comment is after a command on the same line, the Batch SPOC utility sends the entire input string to the Operations Manager (OM) and the comments are parsed by IMS.

The output of the Batch SPOC utility is formatted command responses located in the SYSPRINT file.

If more than one command is issued, the responses appear in the same order as the commands appear in the SYSIN file. The default record length is 133. The command response is formatted in a similar format to the TSO SPOC display. If the records are too long, they wrap to the next line. You can specify DCB information in the JCL or in the data set allocation to allow longer records in the SYSPRINT file.

JCL specifications

The batch SPOC utility is invoked using standard JCL statements.

EXEC statement

The format of the EXEC statement is:

```
//SPOC      EXEC PGM=CSLUSPOC,PARM=('IMSPLEX=plex,ROUTE=sysid,WAIT=05:00,F=option')
```

Parameter keywords

IMSPLEX
Required parameter that specifies the 1- to 5-character suffix of the IMSplex name.

F

Optional parameter that specifies the print format of the SPOC output. You can specify one of the following values:

WRAP
Wraps to the next line as needed. This is the default.

BYCOL
Lines of data are grouped together by the column.

BYRSC
Lines of data are grouped together by the resource.

ROUTE

Optional parameter that specifies the SYSIDs of IMSplex members that are to execute the command. If ROUTE is not specified, all members of the IMSplex will execute the command. If more than one member is specified, enclose the list in parenthesis and separate the names with commas. For example:

```
//  PARM=('IMSPLEX=PLEX1,WAIT=30,ROUTE=(IMSZ,IMSA)')
```

If ROUTE=* is specified, the command is routed to all registered command processing clients in the IMSplex. If ROUTE=% is specified, the command is routed to only one command processing client in the IMSplex that is registered for the command and that has MASTER capability. The Operations Manager chooses the command processing client.

WAIT

Optional parameter that specifies the wait time for individual commands. The wait value is in minutes and seconds (MMM:SS) or just seconds (SSSSS). OM will return a single response as soon as a response is received from all of the members of the IMSplex. If the interval expires, OM will return any responses from IMSplex members, plus an indication that some did not reply. The Batch SPOC utility will wait for each command to complete before issuing the next command. The default wait value is five minutes (5:00). The WAIT time applies to every command in the SYSIN file. The user can specify a wait time of zero seconds; in this case, the batch SPOC issues a command but does not wait for the response.

Return codes

The following return codes are produced:

Code
Meaning

0
The utility completed successfully.

4
Warning messages were issued. Check the output file.

8
A problem was encountered. Check the output file. One or more IMS operator commands failed. Rerun the utility with commands as needed.

12
Internal error. Batch SPOC utility could not process the command or command response. The error message CSLM0142E that is returned has additional information.

Examples of the Batch SPOC utility

These examples show how to use the Batch SPOC utility to create formatted command responses.

JCL example

The following example shows a simple invocation of a sample batch job with multiple commands but the user may call the utility using other valid JCL.

```
//SPOCJOB   JOB ,
//   MSGCLASS=H,NOTIFY=&SYSUID,USER=&SYSUID//*
//SPOC      EXEC PGM=CSLUSPOC,
//  PARM=('IMSPLEX=PLEX1,ROUTE=IMS3,WAIT=30,F=WRAP')
//STEPLIB   DD DISP=SHR,DSN=IMS.SDFSRESL
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
 QRY IMSPLEX SHOW(JOB,TYPE,   +
                  STATUS)

 QRY TRAN NAME(INV1*) SHOW(ALL) /* inventory appl */
/*EOF
```

Output example

If SYSPRINT is a SYSOUT file, the System Display and Search Facility (SDSF) can be used to view batch job output. The following example shows a sample batch job output.

```
============================================================
Log for. . : QRY IMSPLEX SHOW(JOB,TYPE,STATUS)

IMSplex . . . . . : PLEX1
Routing . . . . . : IMS3
Start time. . . . : 2010.132 15:36:28.11
Stop time . . . . : 2010.132 15:36:29.17
Return code . . . : 00000000
Reason code . . . : 00000000
Command master. . : SYS3

IMSplex  MbrName CC  Member  JobName  Type Status
CSLPLEX1 OM1OM   0   USRT002 USRT002  AOP ACTIVE
CSLPLEX1 OM1OM   0   OM1OM   OM1      OM READY,ACTIVE
CSLPLEX1 OM1OM   0   RM1RM   RM1      RM READY,ACTIVE
CSLPLEX1 OM1OM   0   SCI1SC  SCI1     SCI READY,ACTIVE
CSLPLEX1 OM1OM   0   IMS3    IMS3     IMS READY,ACTIVE
CSLPLEX1 OM1OM   0   SYS1    SYS1     IMS READY,ACTIVE
============================================================
```

Output example with no wait time specified

If no wait time, WAIT=0, is specified, the command response is not available and is not printed. The SYSPRINT file only has short summary information for each command. The following example shows a sample batch job output with no response.

```
============================================================
Log for. . : QRY IMSPLEX SHOW(JOB,TYPE,STATUS)

IMSplex . . . . . : PLEX1
Routing . . . . . :
Start time. . . . : 2010.075 15:36:28.11
============================================================
```

Chapter 25. Database Recovery Control utility (DSPURX00)

Use the Database Recovery Control utility to issue commands that build and maintain the RECON data set, add information to the RECON data set, and generate jobs for utilities.

Commands submitted to the Database Recovery Control utility have the same general format. Each command is composed of a verb and a modifier (separated by a period) and then followed by parameters.

Additionally, the Database Recovery Control utility is used to upgrade a RECON data set from an earlier, supported IMS version using the **CHANGE.RECON UPGRADE** command.

Subsections:

- "Restrictions" on page 531
- "Prerequisites" on page 531
- "Requirements" on page 531
- "Recommendations" on page 531
- "Input and output" on page 531
- "JCL specifications" on page 532

Restrictions

Currently, no restrictions are documented for the DSPURX00 utility.

Prerequisites

Currently, no prerequisites are documented for the DSPURX00 utility.

Requirements

Currently, no requirements are documented for the DSPURX00 utility.

Recommendations

Currently, no recommendations are documented for the DSPURX00 utility.

Input and output

The following figure shows the input and output requirements for the Database Recovery Control utility. Notes that describe the input and output in more detail follow the figure.

© Copyright IBM Corp. 1974, 2024

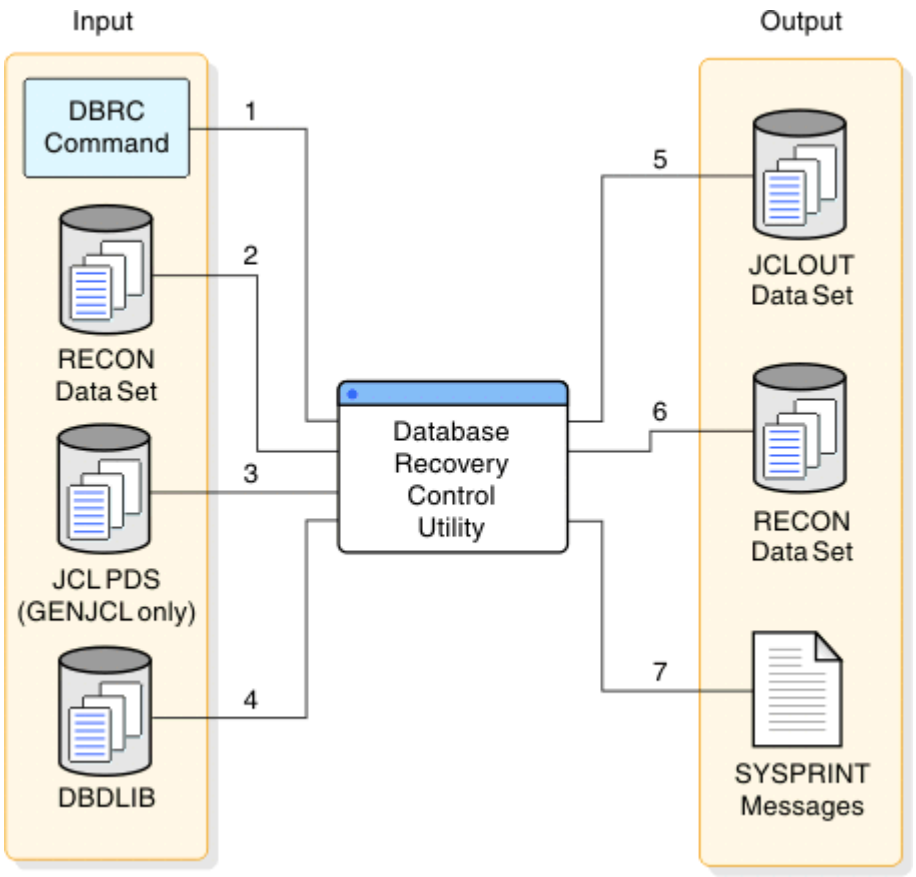


Figure 38. Database Recovery Control utility input and output

Notes to the figure:

1. The DBRC command (input to the Database Recovery Control utility).
2. The RECON data set (input to the Database Recovery Control utility).
3. The PDS, which contains the JCL and control statements for the utility that DBRC uses to generate a job (input to the Database Recovery Control utility).
4. The data set that contains the database descriptions for the databases that are under the control of DBRC (occasional input to the Database Recovery Control utility). If the IMS management of ACBs is enabled, the database descriptions are retrieved from the IMS catalog instead of the DBDLIB data set.
5. Jobs created by **GENJCL** commands (output from the Database Recovery Control utility).
6. The RECON data set, which might be updated by the utility (output from the Database Recovery Control utility).
7. One or more of the following (output from the Database Recovery Control utility):
   - A listing of the input commands.
   - Informational messages associated with their execution or diagnostic messages explaining any failures and return codes.
   - A listing of each job that was created in the case of **GENJCL** commands.

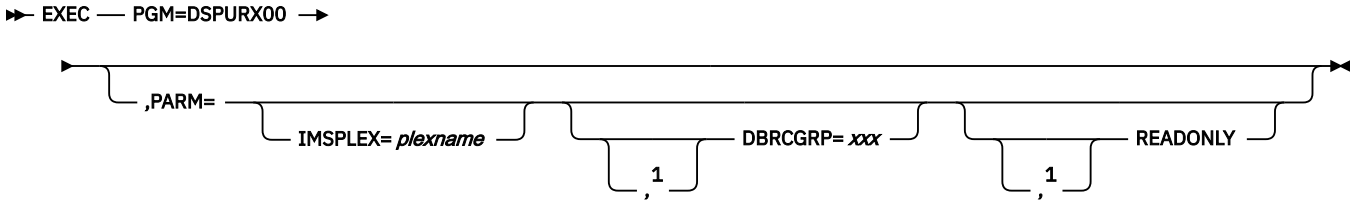
JCL specifications

The Database Recovery Control utility runs as a standard z/OS job.

EXEC statement

Indicates the program to be executed.

You can specify three optional parameters, IMSPLEX, DBRCGRP, and READONLY, in the EXEC statement. If you specify those parameters, it must use the following format:



Notes:

[1] If you specify two or more optional parameters, they must be separated by a comma.

IMSPLEX
Specifies which IMSplex the 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.

READONLY
Specifies that the user has read only authority. Any attempt at updating the statement while in READONLY mode results in an error.

DD statements

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

SYSPRINT
Defines the destination of DBRC diagnostic messages and the listing output. The destination can be a tape or DASD data set, a printer, or it can be routed through the output stream (SYSOUT).

RECON
DD statements for RECON1, RECON2, and RECON3 are omitted so that the RECON data set is allocated dynamically.

IMS
In IMS systems that use ACB, DBD, and PSB libraries, defines the IMS DBDLIB data set. When the IMS management of ACBs is enabled, the IMS DD statement is ignored.

In an IMS system that uses ACB, DBD, and PSB libraries, the following commands require the IMS DD statement:

- **INIT.PART**
- **INIT.DBDS**
- **NOTIFY.REORG**
- If you are initializing a HALDB,**INIT.DB**
- If you are changing a DBDS ddname or area name, **CHANGE.DBDS**
- If you are changing the KEYSTRNG value of a HALDB partition, **CHANGE.PART**

When the IMS management of ACBs is enabled, the preceding commands retrieve database definitions from the IMS catalog that is either registered in the RECON data set or that is specified on the command itself.

JCLPDS, or the DD name you supply with the JCLPDS parameter
Defines the PDS containing skeletal JCL members. It is required only for the **GENJCL** commands.

JCLOUT, or the DD name you supply with the JCLOUT parameter
Defines the data set which is to receive generated JCL. It is required only for the **GENJCL** commands.

SYSIN
Defines the source of input commands. SYSIN can be a tape or DASD data set, a card reader, or it can be routed through the input stream (DD * or DD DATA.)

Related concepts
Data set naming conventions (System Administration)

Examples of the DSPURX00 utility

You can initialize the RECON data set and register one database with two DBDSs.

Inputs and outputs of the Database Recovery Control utility

The following figure is a sample job that initializes the RECON data set and registers one database with two DBDSs.

```
  //INITRCON JOB
  //INIT04   EXEC PGM=DSPURX00
  //STEPLIB  DD   DSN=IMS.SDFSRESL
  //SYSPRINT DD   SYSOUT=A
  //*
  //IMS      DD   DSN=IMS.DBDLIB
  //JCLPDS   DD   DSN=IMS.JCLPDS
  //JCLOUT   DD   DSN=IMS.JCLOUT
  //SYSIN    DD   *
  INIT.RECON SSID(IMS3)
INIT.DB    DBD(DBDESDS1)  SHARELVL(2)
INIT.DBDS  DBD(DBDESDS1)  DDN(DDNESDSA)  GENMAX(3)  -
    REUSE  DSN(IMS.DBDESDS1.DDNESDSA.DSN)    -
    ICJCL(MYIC) RECOVJCL(MYRECOV)          -
INIT.IC    DBD(DBDESDS1)  DDN(DDNESDSA) -
    ICDSN(IMS.*.ICDSN1)
  INIT.IC    DBD(DBDESDS1)  DDN(DDNESDSA) -
    ICDSN(IMS.*.ICDSN2)  ICDSN2(IMS.*.ICDSN2)
  INIT.IC    DBD(DBDESDS1)  DDN(DDNESDSA) -
    ICDSN(IMS.*.ICDSN3)
  //*
INIT.DBDS  DBD(DBDESDS1)  DDN(DDNESDSB)  GENMAX(4)  -
    NOREUSE  DSN(IMS.DBDESDS1.DDNESDSB.DSN)
  //*
INIT.CAGRP  GRPNAME(CAGRP1)  GRPMAX(2) REUSE -
    GRPMEM((DBDESDS1,DDNESDSA),(DBDESDS1,DDNESDSB))
INIT.CA  GRPNAME(CAGRP1) CADSN(IMS.*.CADSN1) -
    VOLLIST(CAVOL1,CAVOL2,CAVOL3)  FILESEQ(4)
  INIT.CA  GRPNAME(CAGRP1) CADSN(IMS.*.CADSN2) -
    VOLLIST(CAVOL4)
  /*
```

Invoking the utility using entry point DSPURXRT

You can use entry point DSPURXRT to specify alternate ddnames for the SYSPRINT, SYSIN, IMS, RECON1, RECON2, and RECON3 data sets.

Use different ddnames if your calling program already uses these ddnames and you want the Database Recovery Control utility to use different data sets.

If there is no RECON1 DD statement, the RECON data set is dynamically allocated. Also, substitution is not allowed for the JCLOUT and JCLPDS DD statements.

Unless stated otherwise, programs that invoke DBRC do not need to pass to DBRC any parameters specified on the EXEC statement. DBRC locates the parameter list passed by z/OS, so the invoking program need not alter this parameter list.

Entry point DSPURXRT must be called in 31-bit mode. DBRC executes in 31-bit mode and returns to the caller in 31-bit mode with the return code set in register 15. Up to two parameters can be passed on the CALL:

- The first parameter specifies the address of an option that can be specified in the PARM parameter of the EXEC statement. The only option supported by the DSPURXRT entry point is READONLY, which specifies whether or not the application needs read only access to the DBRC information.
- The second parameter specifies the address of a list of alternate ddnames for standard data sets that are used during DSPURXRT entry point processing. If standard ddnames are to be used, the second parameter points to either a halfword of binary zeros or is omitted.

Entry point DSPURXRT is invoked by your program through the ATTACH, LINK, or LOAD and CALL macro instructions. Before issuing the invoking macro, the program initializes the appropriate registers and operand list contents. The register contents follow standard linkage conventions:

- Register 1 contains the address of the argument list.
- Register 13 contains the address of a save area.
- Register 14 contains the address of the return point.
- Register 15 contains the address of the entry point DSPURXRT.

The argument list that is pointed to by register 1 consists of up to two pointers to parameters. In the last word in the list, the high order bit is on, indicating that it is the last word. The first word is the address of the options passed to entry point DSPURXRT. The option list consists of a halfword containing the total length of the option list, including this halfword. If no option is to be passed, this length is zero. The IMSPLEX= and DBRCGRP= options are supported only as parameters specified on an EXEC JCL statement.

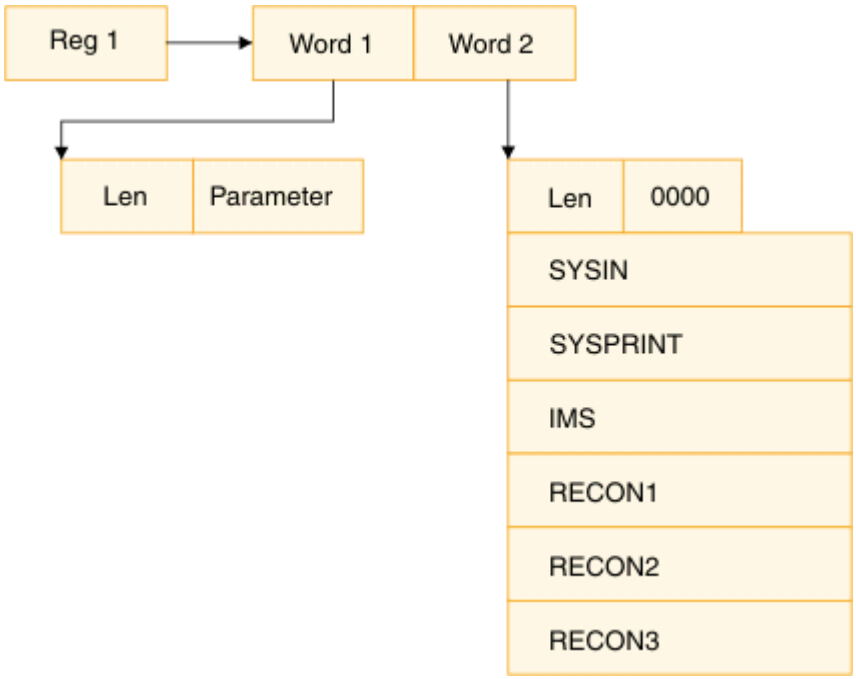


Figure 39. Registers and operand list contents needed to invoke the Database Recovery Control utility

In the figure, the second word is the address of the ddname list. The ddname is left justified and padded on the right with blanks if necessary.

The ddname names are:

- A halfword containing the total length of the ddname list, including this halfword.
- A reserved halfword.
- An eight-byte field containing the ddname to be used in place of the SYSIN data set, or eight bytes of blanks (X'40') if SYSIN is not to be substituted.
- An eight-byte field containing the ddname to be used in place of the SYSPRINT data set, or eight bytes of blanks (X'40') if SYSPRINT is not to be substituted.

- An eight-byte field containing the ddname to be used in place of the IMS data set, or eight bytes of blanks (X'40') if IMS is not to be substituted.
- An eight-byte field containing the ddname to be used in place of the RECON1 data set, or eight bytes of blanks (X'40') if RECON1 is not to be substituted.
- An eight-byte field containing the ddname to be used in place of the RECON2 data set, or eight bytes of blanks (X'40') if RECON2 is not to be substituted.
- An eight-byte field containing the ddname to be used in place of the RECON3 data set, or eight bytes of blanks (X'40') if RECON3 is not to be substituted.

Related reference

z/OS: MVS ATTACH macro

z/OS: MVS CALL macro

z/OS: MVS LINK macro

z/OS: MVS LOAD macro

Chapter 26. Dynamic SVC utility (DFSUSVC0)

Use the Dynamic Supervisor Call (SVC) utility to install an updated version of the IMS Type 2 SVC or DBRC Type 4 SVC without requiring an IPL of the z/OS operating system by changing the z/OS SVC table to point to a new copy of the SVC module.

Subsections:

- "Restrictions" on page 537
- "Prerequisites" on page 537
- "Requirements" on page 537
- "Recommendations" on page 538
- "Input and output" on page 538
- "JCL specifications" on page 538
- "Return codes" on page 539

Restrictions

The following restrictions apply to the Dynamic SVC utility:

- No IMS image (control region, batch, or utility) that uses the IMS Type 2 SVC can be active while attempting to update the Type 2 SVC module. The same restriction does not apply to the DBRC Type 4 SVC module.

Prerequisites

The following prerequisites apply to the Dynamic SVC utility:

- The IMS.SDFSRESL must reflect the correct SVC number to be replaced. You can introduce an error by pointing to the wrong library where a different SVC number (or even non-IMS SVC number) can be associated with this library.

  To specify the SVC numbers so that the numbers are contained in the IMS.SDFSRESL for the DFSUSVC0 utility, you must use the DFSIDEF0 module. To specify a type 2 SVC number for the DFSUSVC0 utility, specify `DFSIDEF  TYPE=PARM,SVC2=`*xxx* on the DFSIDEF macro of the DFSIDEF0 module, where *xxx* is the type 2 SVC number. To specify a type 4 SVC number for the DFSUSVC0 utility, specify `DFSIDEF TYPE=PARM,SVC4=`*yyy* on the DFSIDEF macro of the DFSIDEF0 module, where *yyy* is the type 4 SVC number.

  Warning: Check with your system administrator before using this utility.

Requirements

The following requirements apply to the Dynamic SVC utility:

- The JCL must contain a DFSRESLB DD statement that references an IMS.SDFSRESL.
- The updated SVC module (either IMS Type 2 SVC, DBRC Type 4 SVC, or both) must be in an IMS.SDFSRESL specified on the DFSRESLB DD statement.
- The IMS.SDFSRESL that contains the SVC numbers and the new SVC modules must be an APF-authorized library (standard IMS installation).
- The utility program must reside in an APF-authorized library (usually the IMS.SDFSRESL, but this is not a requirement).

© Copyright IBM Corp. 1974, 2024

- All IMS SVCs must already be defined to z/OS.
- If you use the Dynamic SVC utility to install an updated version of an SVC, you must still add the IMS type 2 SVC module to the z/OS nucleus and add the type 4 SVC module to either SYS1.LPALIB or to an MLPA library. Otherwise, an IPL of z/OS regresses the IMS system to use the old version of the SVC module. In this case, you would need to reinstall the new version of the IMS SVC module by using the Dynamic SVC utility.

Recommendations

Currently, no recommendations are documented for the DFSUSVC0 utility.

Input and output

The input to this utility is either the updated IMS Type 2 SVC module, the updated DBRC Type 4 SVC module, or both. The updated SVC modules must reside in the library that is pointed to by the DFSRESLB DD statement.

The utility determines which SVCs to update and dynamically changes the z/OS SVC table to point to the new SVC modules.

JCL specifications

The Dynamic SVC utility is executed as a standard z/OS job. You must supply the following:

- A JOB statement
- An EXEC statement
- DD statements that define inputs

EXEC statement

This utility runs as a z/OS job.

The EXEC statement must be in one of the following forms:

- `//STEP001 EXEC PGM=DFSUSVC0` or
- `//STEP001 EXEC PGM=DFSUSVC0,PARM='SVCTYPE=(2)'` or
- `//STEP001 EXEC PGM=DFSUSVC0,PARM='SVCTYPE=(4)'` or
- `//STEP001 EXEC PGM=DFSUSVC0,PARM='SVCTYPE=(2,4)'`

The EXEC statement allows you to specify whether the IMS Type 2 SVC module, the DBRC Type 4 SVC module, or both are to be updated. When SVCTYPE=(2) is specified, the IMS Type 2 SVC module is updated. When SVCTYPE=(4) is specified, the DBRC Type 4 SVC is updated. When SVCTYPE=(2,4) is specified, both the IMS Type 2 SVC and the DBRC Type 4 SVC module are updated. If a value is not specified for the SVCTYPE= parameter, the IMS Type 2 SVC module is updated by default.

DD statements

STEPLIB DD

Points to an authorized library that contains the actual DFSUSVC0 utility. The authorized library should be in your IMS.SDFSRESL).

```
//STEPLIB  DD  DSN=SOME.APF.AUTHORIZED.DATASET,DISP=SHR
```

DFSRESLB DD
Points to an authorized library that contains the updated SVC modules and the IMS Type 2 and DBRC Type 4 SVC numbers.

```
//DFSRESLB DD  DSN=SOME.IMS.SDFSRESL,DISP=SHR
```

Return codes

The following return codes are produced:

Code
Meaning

0
Dynamic installation was successful. All specified SVC routines were successfully updated.

8
The installation of at least one of the specified SVC routines failed.

Related tasks
Installing the type 2 SVC module (System Administration)

Examples of the DFSUSVC0 utility

These examples show how to use the DFSUSVC0 utility to replace the IMS Type 2 SVC and the DBRC Type 4 SVC.

The following figures show the JCL needed to replace the IMS Type 2 SVC and the DBRC Type 4 SVC.

Example for replacing IMS type 2 SVC

```
//SVCINIT  JOB MSGLEVEL=1,TIME=1440
//STEP001 EXEC PGM=DFSUSVC0,PARM='SVCTYPE=(2)'
//STEPLIB  DD  DSN=SOME.APF.AUTHORIZED.DATASET,
//             DISP=SHR
//DFSRESLB DD  DSN=SOME.IMS.SDFSRESL,
//             DISP=SHR
```

Example for replacing DBRC type 4 SVC

```
//SVCINIT  JOB MSGLEVEL=1,TIME=1440
//STEP001 EXEC PGM=DFSUSVC0,PARM='SVCTYPE=(4)'
//STEPLIB  DD  DSN=SOME.APF.AUTHORIZED.DATASET,
//             DISP=SHR
//DFSRESLB DD  DSN=SOME.IMS.SDFSRESL,
//             DISP=SHR
```

Example for replacing both SVC modules

```
//SVCINIT  JOB MSGLEVEL=1,TIME=1440
//STEP001 EXEC PGM=DFSUSVC0,PARM='SVCTYPE=(2,4)'
//STEPLIB  DD  DSN=SOME.APF.AUTHORIZED.DATASET,
//             DISP=SHR
//DFSRESLB DD  DSN=SOME.IMS.SDFSRESL,
//             DISP=SHR
```

Chapter 27. Global Online Change utility (DFSUOLC0)

Use the Global Online Change utility to initialize, recreate, or unlock the OLCSTAT data set.

To use the Global Online Change utility to initialize OLCSTAT data sets, you must first define either MDBS=A or MDBS=B, even for an IMS that does not define the MODBLKS data sets. For an IMSplex to be enabled for global online change, the Global Online Change utility must be used to initialize the OLCSTAT before the first IMS in the IMSplex cold starts the first time. The Global Online Change utility can be used to recreate the OLCSTAT data set after an error that renders the OLCSTAT data set unusable.

The Global Online Change commands, **INITIATE OLC PHASE(PREPARE)**, followed by **INITIATE OLC PHASE(COMMIT)**, cause the inactive library to become the active library.

The OLCSTAT data set contains the global online change status, which includes the modify id, the active online change libraries, a lock field, the last online change, and a list of IMS systems that are current with the online change libraries. When an IFP region is running, OLC commit stops because of existing active route code. Therefore, all IFP regions must be terminated before commit.

The Online Change Copy utility supports an OLCSTAT DD statement, to identify the global online change status data set name. The OLCSTAT data set is comparable to the MODSTAT data set used by local Online Change.

Attention: Use the recreate and unlock functions with extreme caution. Use the unlock function only if a series of errors has left the OLCSTAT data set locked and no online change is in progress. If you inadvertently destroy valid OLCSTAT data set contents, global online change and initialization of additional IMS systems fail until the OLCSTAT data set is re-initialized.

Establish an OLCSTAT data set recovery procedure to deal with the loss of the OLCSTAT data set. After every successful global online change, record the following data:

- The modify id
- The active online change library suffixes
- The list of IMS systems that are current with the online change libraries

If the OLCSTAT data set is destroyed, run the initialize function of the Global Online Change utility with the recorded data to re-initialize the OLCSTAT data set.

The DFS3499 message, which identifies the current values of the online change libraries in the OLCSTAT data set, follows the DFS994 checkpoint message. The DFS3410 message at initialization also identifies the current online change libraries from the OLCSTAT data set.

DFSUOLC0 can also be used to update the IMS version in the OLCSTAT data set header. When you upgrade to a new version of IMS that is higher than the previous level, the IMS version in the OLCSTAT header will be automatically updated during IMS cold start. However, this automatic update will not be completed if you fall back to a lower IMS level. In this case, the Global Online Change utility must be run to update the IMS version in the OLCSTAT header to the lower IMS level.

If you use the UNL function to unlock the OLCSTAT data set, and all of the IMSs failed during a global online change, you will need to start IMS. IMS will start because the OLCSTAT data set is no longer locked.

If you use the UNL function to unlock the OLCSTAT data set after the global online change command master failed during online change while holding the OLCSTAT data set lock, the next global online change command to issue depends upon the state of the online change.

- If the global online change passed the commit 1 phase where the OLCSTAT data set has been updated, the online change is committed and the only command that can be issued is another **INIT OLC PHASE(COMMIT)** command to complete the commit by cleaning up.
- If the global online change failed before commit phase 1 where the OLCSTAT data set is updated, the global online change is not committed. Commit can be attempted again with an **INIT OLC PHASE(COMMIT)** command, or the global online change can be aborted with a **TERMINATE OLC** command.

© Copyright IBM Corp. 1974, 2024

To see the state of the global online change, issue the **QUERY MEMBER TYPE(IMS) SHOW(STATUS)** command to see the state of all the IMSs participating in the global online change. For more information, see Handling errors for global online change commands (System Administration).

Subsections:

- "Restrictions" on page 542
- "Prerequisites" on page 542
- "Requirements" on page 542
- "Recommendations" on page 542
- "JCL specifications" on page 542

Restrictions

Currently, no restrictions are documented for the DFSUOLC0 utility.

Prerequisites

Currently, no prerequisites are documented for the DFSUOLC0 utility.

Requirements

Currently, no requirements are documented for the DFSUOLC0 utility.

Recommendations

Currently, no recommendations are documented for the DFSUOLC0 utility.

JCL specifications

The following JCL will run with the DFSUOLC procedure and invoke the utility with VERS=2, which is the default.

```
//DFSUOLC0 JOB
//STEP1 EXEC DFSUOLC,FUNC=,ACBS=,MDBS=,FMTS=,MDID=,PLEX=
//SYSIN DD *
//
```

The following JCL will run with the DFSUOLC procedure and invoke the utility with VERS=1.

```
//DFSUOLC0 JOB
//STEP1 EXEC DFSUOLC,FUNC=,ACBS=,MDBS=,FMTS=,MDID=,PLEX=,VERS=1
//SYSIN DD *
//
```

Procedure statement

The JCL shown in the following figure shows the statements used to invoke the DFSUOLC procedure. The procedure is built during SMP/E processing and placed in the ADFSPROC and SDFSPROC data sets.

```
//PROC FUNC=,ACBS=,MDBS=,FMTS=,MDID=,PLEX=,VERS=,SOUT=A,
//       OLCPLEX=IMSPLEX,
//       NODE2=IMS
//STEP1  EXEC PGM=DFSUOLC0,
//              PARM=(&FUNC,&ACBS,&MDBS,&FMTS,&MDID,&PLEX,&VERS)
//STEPLIB DD DSN=&NODE2..SDFSRESL,DISP=SHR
//OLCSTAT DD &OLCPLEX..OLCSTAT,DISP=OLD
```

```
//SYSPRINT DD SYSOUT=&SOUT
//SYSIN DD DUMMY
```

EXEC statement

The format of the EXEC statement is:

```
PGM=DFSUOLC0,PARM=(&FUNC,&ACBS,&MDBS,&FMTS,&MDID,&PLEX,&VERS)
```

The Global Online Change utility (DFSUOLC0) supports the following parameters.

ACBS
Specifies the IMS JCL IMSACB DD statement suffix for the active ACB library. The suffix can be A or B. A means IMSACBA is the DD statement of the active library. B means IMSACBB is DD statement of the active library.

FMTS
Specifies the IMS JCL FORMAT DD statement suffix for the active MFS FORMAT library. The suffix can be A or B. A means FORMATA is the DD statement of the active library. B means FORMATB is the DD statement of the active library. FORMAT contains online MFS definitions to be used as the format library by the online system. MFS-supported terminals and the MFS Language utility program require their use. This parameter is required, even if no IMS in the IMSplex uses the MFS format library.

FUNC
Specifies the Global Online Change utility function to perform.

ADD
Add one or more IMS members to the list of IMS systems that are current with the online change libraries. Add an IMS when the OLCSTAT data set suffered an error that made it unusable and you are trying to recreate the OLCSTAT data set contents. The IMS systems to add must be specified with the SYSIN DD card.

Add IMS systems that are current with the online change libraries; for example, IMS systems that are currently up.

Attention: If you add an IMS that is not current with the online change libraries, and warm start that IMS, the warm start might fail.

DEL
Delete one or more IMS systems from the list of IMS systems that are current with the online change libraries.

Delete an IMS when you never intend to bring the IMS up again, so that the **INITIATE OLC** command does not need to be specified with the FRCABND or FRCNRML keyword. The IMS systems to delete must be specified with the SYSIN DD card.

INI
Function to initialize the OLCSTAT data set. ACBS, MDBS, FMTS, and MDID must also be specified. An optional list of one or more IMS systems can be specified with the SYSIN DD statement. If no IMS systems are specified with the SYSIN DD statement, the list of IMS systems is deleted from the OLCSTAT data set.

The INI function is required before the first IMS in the IMSplex cold starts the first time to initialize the OLCSTAT data set.

If the OLCSTAT record contents are lost and must be reconstructed, you must run the Global Online Change utility INI function to construct its contents with the correct values for the online change identifier and online change library ddnames. You might also want to add IMS systems that are current with the online change libraries using the SYSIN DD statement. Keep track of the current online change libraries and modify id so that you can reconstruct the OLCSTAT data set contents in case of failure.

UNL
Function to reset the OLCSTAT data set lock after all IMS systems failed during online change.

The UNL function of the Global Online Change utility is required to reset the OLCSTAT data set lock, in the case where all IMS systems in the IMSplex failed during an online change. Online change sets a lock field in the OLCSTAT data set to prevent other IMS systems from initializing during the online change. IMS initialization fails if a global online change is in progress (between the prepare and commit phases), because the OLCSTAT data set lock is set. When an IMS tries to initialize after all IMS systems failed during online change, IMS initialization is rejected because the OLCSTAT data set lock is set. In this case, you must run the Global Online Change utility with the UNL function to reset the OLCSTAT data set lock. No IMS can initialize until the OLCSTAT data set lock is reset. The UNL function should rarely need to be used. It is needed only if all the IMS systems fail during an online change.

MDBS

Specifies the IMS JCL MODBLKS DD statement suffix for the active MODBLKS data set. The suffix can be A or B. A means MODBLKSA is the DD statement of the active library. B means MODBLKSB is the DD statement of the active libraries.

If DRD is enabled, you can start IMS without defining MODBLKS DD cards. However, to use the Global Online Change utility to initialize OLCSTAT data sets, you must first define either MDBS=A or MDBS=B, even for an IMS that does not define the MODBLKS data sets. If the MODBLKS data sets are not defined to IMS, the MODBLKS value in the OLCSTAT data set is ignored.

MDID

Specifies the modifyid (online change status identifier) for the INI function. This should be initialized to zero to indicate that the number of global online changes performed is zero. The modifyid is used to determine whether an IMS was down for one or more online changes and to determine the kind of restart IMS can perform. The modifyid is used by IMS internal processing:

- To determine whether IMS must cold start.

  If an IMS participated in the last global online change, its modifyid matches the modifyid in the OLCSTAT data set. This IMS is allowed to warm start. If an IMS did not participate in the last global online change, its modifyid does not match the modifyid in the OLCSTAT data set. It is permitted to warm start if its restart type does not conflict with the last online change that was performed. If the IMS was down for two or more global online changes, it must cold start.
- To recover security status during emergency restart processing.

PLEX

Specifies a 1-5 character identifier that specifies the z/OS cross-system coupling facility CSL IMSplex group name for the UNL function. PLEX is required for the UNL function. All OM, RM, SCI, IMS, and ODBM IMSplex members that are in the same IMSplex sharing group, sharing either databases or message queues, must specify the same identifier. The same identifier must also be used for the IMSPLEX= parameter in the CSLSIxxx, CSLOIxxx, CSLRIxxx and DFSCGxxx PROCLIB members.

SOUT

Specifies the class assigned to SYSOUT DD statements.

The STEPLIB DD statement identifies the IMS.SDFSRESL. The IMS.SDFSRESL contains the IMS required modules. This IMS.SDFSRESL must be the highest level available in the IMSplex.

The SYSUDUMP DD statement defines the dump data set for this program.

The SYSPRINT DD statement defines the message output data set.

The OLCSTAT DD statement identifies the OLCSTAT (global online change status) data set name. The OLCSTAT DD statement is required.

The SYSIN DD statement contains the list of IMS systems to define, add, or delete. Specify one IMS ID per line.

The SYSIN DD statement specified with the ADD function adds one or more IMS systems to the existing list of IMS systems in the OLCSTAT data set.

The SYSIN DD statement specified with the DEL function deletes one or more IMS systems from the existing list of IMS systems in the OLCSTAT data set.

The SYSIN DD statement specified with the INI function defines a new list of IMS systems. If IMS records already existed, they are wiped out.

VERS
Specifies the OLCSTAT version that is used to initialize the OLCSTAT. The valid values are 1 and 2. The default value is 2.

1
Indicates that the OLCSTAT is to be initialized to Version 1 format for the header.

Use VERS=1 if there are one or more IMS Version 9 or IMS Version 10 systems in the OLCSTAT along with IMS 15.5 systems.

2
Indicates that the OLCSTAT is to be initialized to Version 2 format for the header.

Use VERS=2 if all IMS systems in the OLCSTAT are IMS 15.5. VERS=2 is required to utilize the TYPE(ACBMBR) member online change.

Related reference
"Online Change Copy utility (DFSUOCU0)" on page 573
Use the Online Change Copy utility (DFSUOCU0) as one step in the process of preparing an IMS or an IMSplex for a local or global online change.

Examples of the DFSUOLC0 utility

These examples show how to use the DFSUOLC procedure to invoke the DFSUOLC0 utility to initialize the OLCSTAT data set.

Parameters

You can specify the following execution parameters in the DFSUOLC procedure:

- NODE2
- OLCPLEX: specifies the data set qualifier or qualifiers for the OLCSTAT data set, which corresponds to the IMSPLEX parameter of the IMSGEN macro.

NODE2 is described in Parameter descriptions for IMS procedures (System Definition)

Global Online Change utility example 1

The following example shows the JCL for the Global Online Change utility to initialize the OLCSTAT data set before the first IMS cold starts the first time.

```
//DFSUOLC0 JOB
//STEP1 EXEC DFSUOLC,FUNC=INI,ACBS=A,MDBS=A,FMTS=A,MDID=0
//SYSIN DD *
/*
//
```

Global Online Change utility example 2

The following example shows the JCL for the Global Online Change utility that initializes the OLCSTAT data set header. You should rarely need to include a list of IMS systems when initializing the OLCSTAT data set header. For example, if the OLCSTAT data set became unusable, you would have to initialize the OLCSTAT header. If you know which IMS systems are current with the online change libraries, you could include those IMS systems in the list. If IMSIDs are not specified, no IMSID will be listed on the OLCSTAT data set record.

```
//DFSUOLC0 JOB
//STEP1 EXEC DFSUOLC,FUNC=INI,ACBS=A,MDBS=A,FMTS=A,MDID=0
//SYSIN DD
IMSA
IMSB
```

```
/*
//
```

Global Online Change utility example 3

The following example shows the JCL for the Global Online Change utility to initialize the OLCSTAT data set to Version 1 before the IMS cold starts the first time.

```
//DFSUOLC0 JOB
//STEP1 EXEC DFSUOLC,FUNC=INI,ACBS=A,MDBS=A,FMTS=A,MDID=0,VERS=1
//SYSIN DD *
//
```

Global Online Change utility example 4

The following example shows the JCL for the Global Online Change utility that initializes the OLCSTAT data set header to Version 1 and add the list of IMS 15.5 systems that are current with the online change libraries.

```
//DFSUOLC0 JOB
//STEP1 EXEC DFSUOLC,FUNC=INI,ACBS=A,MDBS=A,FMTS=A,MDID=0,VERS=1
//SYSIN DD
IMSA
IMSB
//
```

Chapter 28. MFS Service utility (DFSUTSA0)

Use the MFS Service (MFSRVC) utility (DFSUTSA0) to control and maintain MFS intermediate text blocks and control blocks after they have been processed and stored by the MFS Language utility (DFSUPAA0).

An intermediate text block (ITB) is a message, format, partition set, or table source language definition that is stored in the IMS.REFERAL library. A control block is a message or format definition that is stored in the IMS.FORMAT, IMS.FORMATA, IMS.FORMATB, or IMS.TFORMAT library.

The service utility performs the following functions:

- INDEX creates a special directory for faster access to IMS.FORMAT control blocks.
- DELETE deletes specified contents of the special index directory ($$IMSDIR).
- SCRATCH scratches specified contents of the IMS.FORMAT and IMS.REFERAL libraries and their directories (SCRATCH also operates on IMS.TFORMAT).
- RELATE produces an interpreted listing of the contents of the IMS.REFERAL library.
- LIST produces an interpreted listing of either:
  - The contents of the IMS.FORMAT or IMS.TFORMAT library
  - The contents of the special index directory in the IMS.FORMAT library
  - The contents of the MFS device characteristics table (DFSUDT0x) in the IMS.SDFSRESL library.

Subsections:

- “Restrictions” on page 547
- “Prerequisites” on page 548
- “Requirements” on page 548
- “Recommendations” on page 548
- “Input and output” on page 548
- “JCL specifications” on page 548
- “Utility control statements” on page 549
- “INDEX function description” on page 553
- “DELETE function description” on page 553
- “SCRATCH function description” on page 554
- “RELATE function description” on page 556
- “LIST function description” on page 557
- “Function descriptions parameters” on page 558
- “LIST the MFS device characteristics table” on page 559
- “LIST DEVCHAR output” on page 559
- “Return codes” on page 560

Restrictions

The following restrictions apply to this utility:

- Do not run the MFS Service utility concurrently with the MFS Language utility (MFSUTL procedure) if they are accessing the same data set.
- Do not run the MFS Service utility concurrently with the IMS online control region if they are both accessing the active format library.

© Copyright IBM Corp. 1974, 2024

Prerequisites

Currently, no prerequisites are documented for the DFSUTSA0 utility.

Requirements

Currently, no requirements are documented for the DFSUTSA0 utility.

Recommendations

Currently, no recommendations are documented for the DFSUTSA0 utility.

Input and output

Currently, there are no input or output statements for DFSUTSA0.

JCL specifications

The DFSUTSA0 utility requires a procedure statement, the EXEC statement, and the DD statements.

The following figure shows a one-step procedure for maintaining the MFS libraries. The procedure is built during SMP/E processing and placed in the ADFSPROC and SDFSPROC data sets.

```
//        PROC DEVCHAR=0,SYS2=,SOUT=A,
//                 NODE1=IMS,
//                 NODE2=IMS
//MFSRVC  EXEC PGM=DFSUTSA0,REGION=4M,PARM='DEVCHAR=&DEVCHAR'
//STEPLIB  DD DSN=&NODE2..&SYS2.SDFSRESL,DISP=SHR
//*
//*        PRINT FILES
//*
//SYSPRINT DD SYSOUT=&SOUT
//*           DCB=(RECFM=VBA,LRECL=137)
//SYSSNAP  DD SYSOUT=&SOUT
//*           DCB=(RECFM=VBA,LRECL=125,BLKSIZE=1632)
//SYSUDUMP DD SYSOUT=&SOUT
//*
//*        REFERAL LIBRARY
//*
//REFIN    DD DSN=&NODE1..&SYS2.REFERAL,DISP=OLD
//*
//*        ON-LINE FORMAT LIBRARY
//*
//FORMAT   DD DSN=&NODE1..&SYS2.FORMAT,DISP=SHR
//*
//*
//*         //SYSIN DD * MUST BE SUPPLIED BY
//*         USER WITH INPUT CONTROL CARD STREAM
//*
//*         ALL DISP=OLD SPECIFICATIONS OF THIS
//*         PROCEDURE ARE REQUIRED .......
//*
//*
```

Procedure statement

The procedure statement must be in the form:

```
DEVCHAR=0,SYS2=,SOUT=A
```

DEVCHAR=
Specifies the device characteristics table. The default is 0.

SOUT=
Specifies the SYSOUT class. The default is A.

SYS2=
Specifies an optional second level dsname qualifier for those data sets which are designated as "Optional Replicate" in an XRF complex. When specified, the operand must be enclosed in quotes and must include a trailing period; for example, SYS2='IMSA.'.

Invoking the procedure example

The JCL statements to invoke the MFSRVC procedure are shown in the following example.

```
//MFSRVC   JOB  MSGLEVEL=1
//        EXEC  MFSRVC
//SYSIN   DD     *
     END
/*
```

SYSIN DD
Defines the input control statement data sets.

EXEC statement

The EXEC statement must be in the form:

```
PGM=DFSUTSA0,REGION=250K,PARM='DEVCHAR=&DEVCHAR'
```

REGION=
Specifies the region size for the execution of the MFS Service utility. The default is 4 MB.

PARM=
The PARM= field must be in the form:

```
PARM='DEVCHAR=&DEVCHAR'
```

where &DEVCHAR is the device characteristics table to be listed.

DD statements

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

SYSPRINT DD
Defines the output message data set. It can be a printer or it can be routed through the output stream. If DISP=(MOD,...) is specified, it can be a tape volume or a direct-access device. The output can be blocked as a multiple of 121.

SYSIN
Refers to the input data set, which can be a sequential data set or a member of a partitioned data set.

SYSSNAP
Refers to a data set that is used to receive the output from a SNAP macro if certain severe errors are detected.

Utility control statements

The control statements used by the MFS Service utility program utilize the same syntax and many of the same keywords as the source statement input to the preprocessor.

Exception: An exception to this occurs when comments are placed on a statement or portion thereof.

Requirement: Because of the extreme range of allowable operations with a FMT block or blocks, the utility requires that comments on a statement be started with /* (for example, LIST ALL /* THIS STATEMENT...).

Positional parameters

ALL
Where allowed, implies invocation of all functions supported for the associated operator. For example, INDEX ALL implies the insertion of all MID, MOD, DIF, and DOF names that exist in IMS.FORMAT into the special index directory ($$IMSDIR) to be used by the online control region for direct block access.

PDIR
Implies invocation of the associated operation against the PDS directory entries of IMS.FORMAT or, when used with LIST or SCRATCH, the PDS directories of IMS.TFORMAT. For example, LIST PDIR causes the contents of IMS.FORMAT or IMS.TFORMAT to be listed in interpreted format.

INDEX
Directs the invocation of an operation to the special index directory ($$IMSDIR) used by the online control region.

REFER
Directs the invocation of an operation to IMS.REFERAL used by the MFS Language utility as an historical intermediate text storage library.

FORMAT
Directs the invocation of an operation to IMS.FORMAT, or, when used with LIST or SCRATCH, to IMS.TFORMAT.

DEVCHAR
Valid only with the LIST function. It causes the device characteristics table identified by the suffix (DEVCHAR=suffix) in the EXEC parameter to be printed. If the EXEC parameter is not specified, the contents of DFSUDT00 are printed.

Keywords

MSG=*name* | ALL
Directs the invocation of a function

name
Directs the invocation of a function to a specific message control block, MID or MOD. Name must be specified as a 1- to 8-character alphanumeric value, the first character of which must be alphabetic.

ALL
Directs the invocation of an operation to all message descriptors.

FMT=*name* | ALL
Directs the invocation of an operation

name
Directs the invocation of an operation to a specific device format.

ALL
Directs the invocation of an operation to all device formats. Unless further qualified, the operation proceeds against all FMT control blocks in IMS.FORMAT.

The FMT control block can consist of multiple FMT control blocks (DIFs and the DOFs) in IMS.FORMAT and, unless further qualified, the operation proceeds against all FMT control blocks with the same root name. You can specify "name" as a 1- to 6-character alphanumeric value, the first character of which must be alphabetic.

TBL=name
Directs the SCRATCH function to scratch a TBL ITB from the IMS.REFERAL library. The keyword is valid only on the SCRATCH utility control statement and only if the REFER positional parameter is

specified. The names of all the TBLs that reside in IMS.REFERAL can be obtained through the Service Utility RELATE function.

PDB=name
Directs the function to scratch a PDB ITB from the IMS.REFERAL library. The keyword is valid only on the SCRATCH utility control statement and only if the REFER positional parameter is specified. The names of all the PDBs that reside in IMS.REFERAL can be obtained through the Service Utility RELATE function.

DEVCHAR=*x*
Causes the device characteristics table identified by the suffix following the "=" to be printed. This parameter is only valid with the LIST function.

DEV=
Qualifies a specified FMT control block as applying either to a particular device, a secondary logic unit type, or a remote program or to all (ALL).

Specify
Device

3270
3270 or SLU 2 display station

3270-A
For all 3270 or SLU 2 display stations that have been defined during IMS system definition using the device symbolic name.

3270-A1,...,A15
For a single 3270 or SLU 2 display station that has been defined during IMS system definition using the specific device symbolic name.

3270P
3270 printer

FIN
Finance application program

FIDS
Finance display component (6x40) (for example, 3604-1 or 3604-2)

FIDS3
Finance display component (12x40) (for example, 3604-3)

FIDS4
Finance display component (16x64) (for example, 3604-4)

FIDS7
Finance display component (24x80) (for example, 3604-7)

FIJP
Finance journal printer

FIPB
Finance passbook printer

FIFP
Finance administrative printer

DPM-A
For all SLU P devices that have been defined during system definition using this device symbolic name.

DPM-B
For all ISC nodes that have been defined during system definition using this device symbolic name.

DPM-A1,...,A15
SLU P

DPM-B1,...,B15
ISC nodes

SCS1
The following console keyboard/printers: 3767; NTO; 3771; 3773; 3774; 3775; 3776; 3777; and SLU 1 (print data set) or SLU 4.

SCS2
3521 card punch, 3501 card reader, 2502 card reader; and SLU 1 (transmit data set) or, SLU 4

ALL
All specified devices

MDL=1 | 2 | ALL
Determines FMT control block operation.

1
Restricts a FMT control block operation to control blocks for Model 1 3270/3270P stations.

2
Restricts a FMT control block operation to control blocks for Model 2 3270/3270P stations.

ALL
Directs FMT control block operation to control blocks for both Model 1 and Model 2 3270/3270P stations.

This keyword applies only to 3270 and 3270P devices.

DIV=INPUT | OUTPUT | INOUT
Determines FMT control block operation.

INPUT
Restricts a FMT control block operation to DIFs.

OUTPUT
Restricts a FMT control block operation to DOFs.

INOUT
Indicates a FMT control block operation is to proceed for both DIFs and DOFs.

FEAT=IGNORE | n[*nn*] | (list) | ANY
Determines FMT control block operation.

IGNORE
Restricts a FMT control block operation to control blocks for which FEAT=IGNORE was specified on the DEV statement to the MFS Language utility.

n[*nn*]
With either:

- A print line of 120, 126, or 132
- User-defined features 1—10

(list)
Restricts a FMT control block operation to control blocks with a specific feature combination. The specifications allowed for "list" are as follows.

For DEV=FIFP:

```
DUAL
132
(DUAL,132)
```

For DEV=3270 or DEV=3270-An:

```
PEN      ,PFK      ,CARD
NOPEN    DEKYBD    NOCD
         NOPFK
```

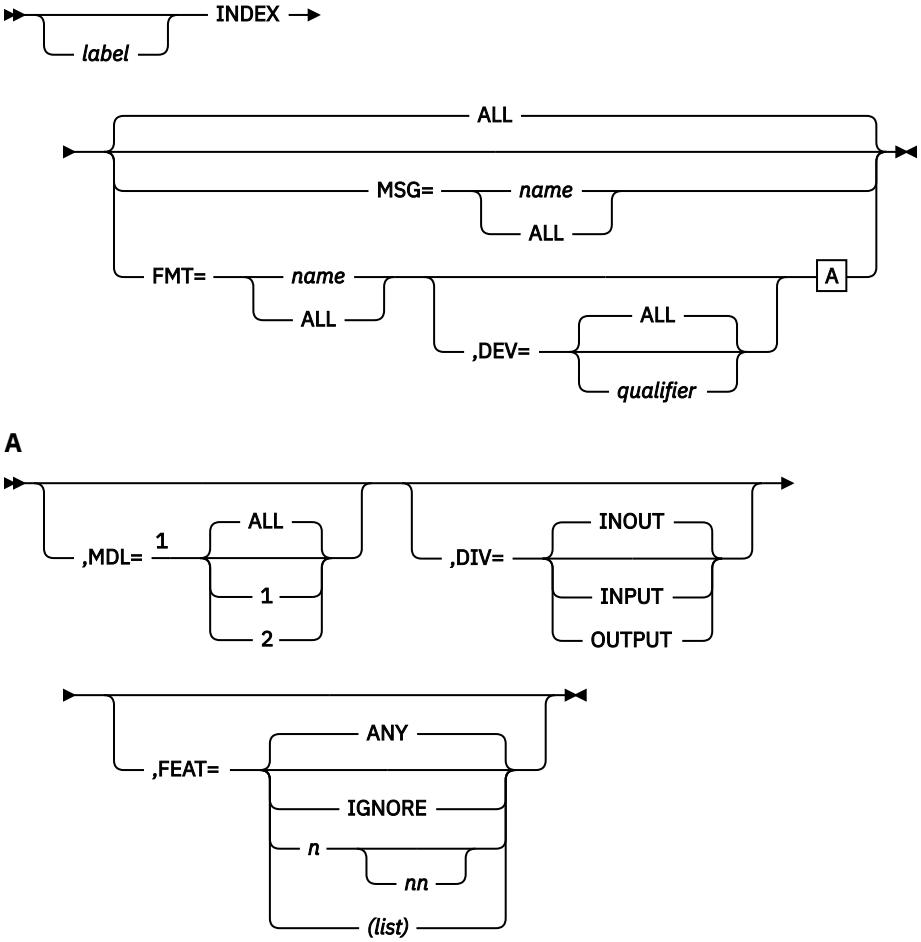
Enter commas only where required to separate specifications that are actually coded. Feature specifications do not depend on position. You must code at least one alternative. The same

feature value results, whether one, two, three, or none of the NOPEN, NOPFK, or NOCD parameters are specified.

ANY
Directs FMT control block operation to all control blocks without restrictions on the feature specification.

INDEX function description

The INDEX function places the specified names of control blocks (or sets of related control blocks) in a special index directory ($$IMSDIR), that is used to provide quick online access to the control blocks.



Notes:

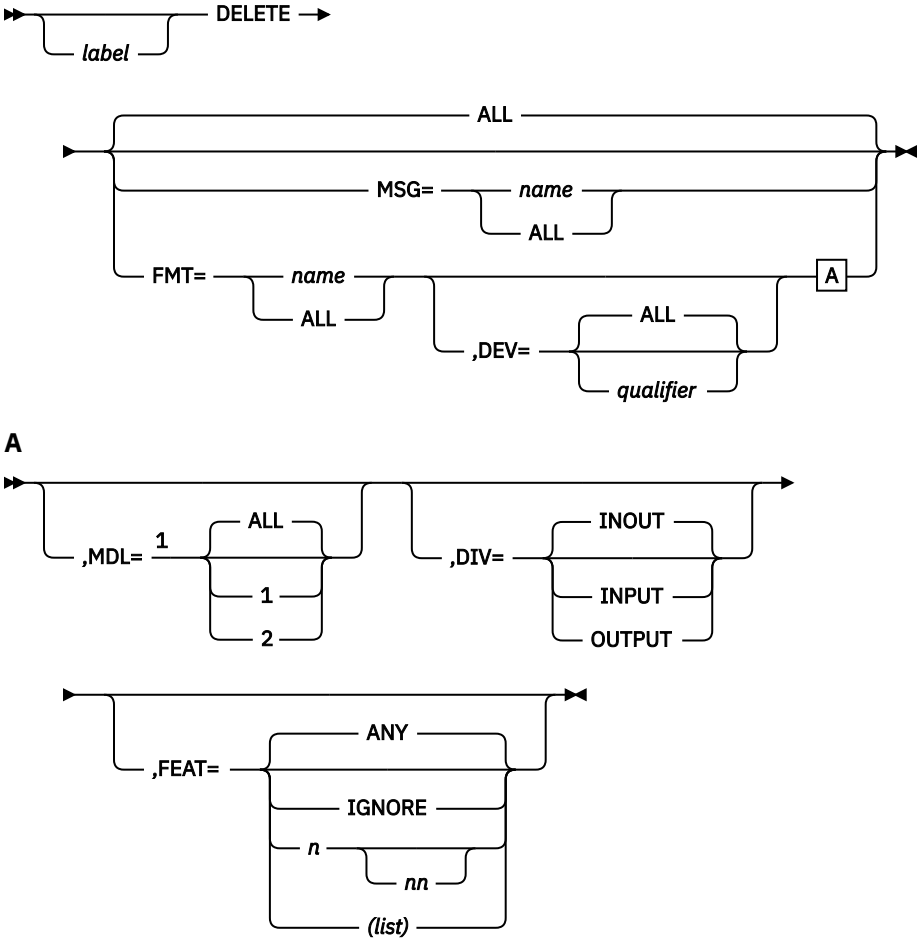
[1] MDL=only applies to 3270 and 3270P devices.

If $$IMSDIR has been created by the INDEX function, it is read into the MFS buffer pool during initialization of the online IMS control region and made permanently resident there. If a requested control block is not found in the MFS buffer pool, the MFS pool buffer manager next looks for an entry in $$IMSDIR. The entry, if found, allows the MFS pool manager to issue a direct read for the control block.

Using the special index directory to replace two direct-access storage reads with one is a performance advantage, but this advantage must be weighed against the storage cost in the online control region (14 bytes per control block indexed in the MFS buffer pool). Indexing only the most frequently used control blocks, which might be a small percentage of the total, can be advisable.

DELETE function description

The DELETE function specifies the names of a control block, or a set of control blocks, which are to be deleted from the special index directory ($$IMSDIR) used by the online MFS pool manager.

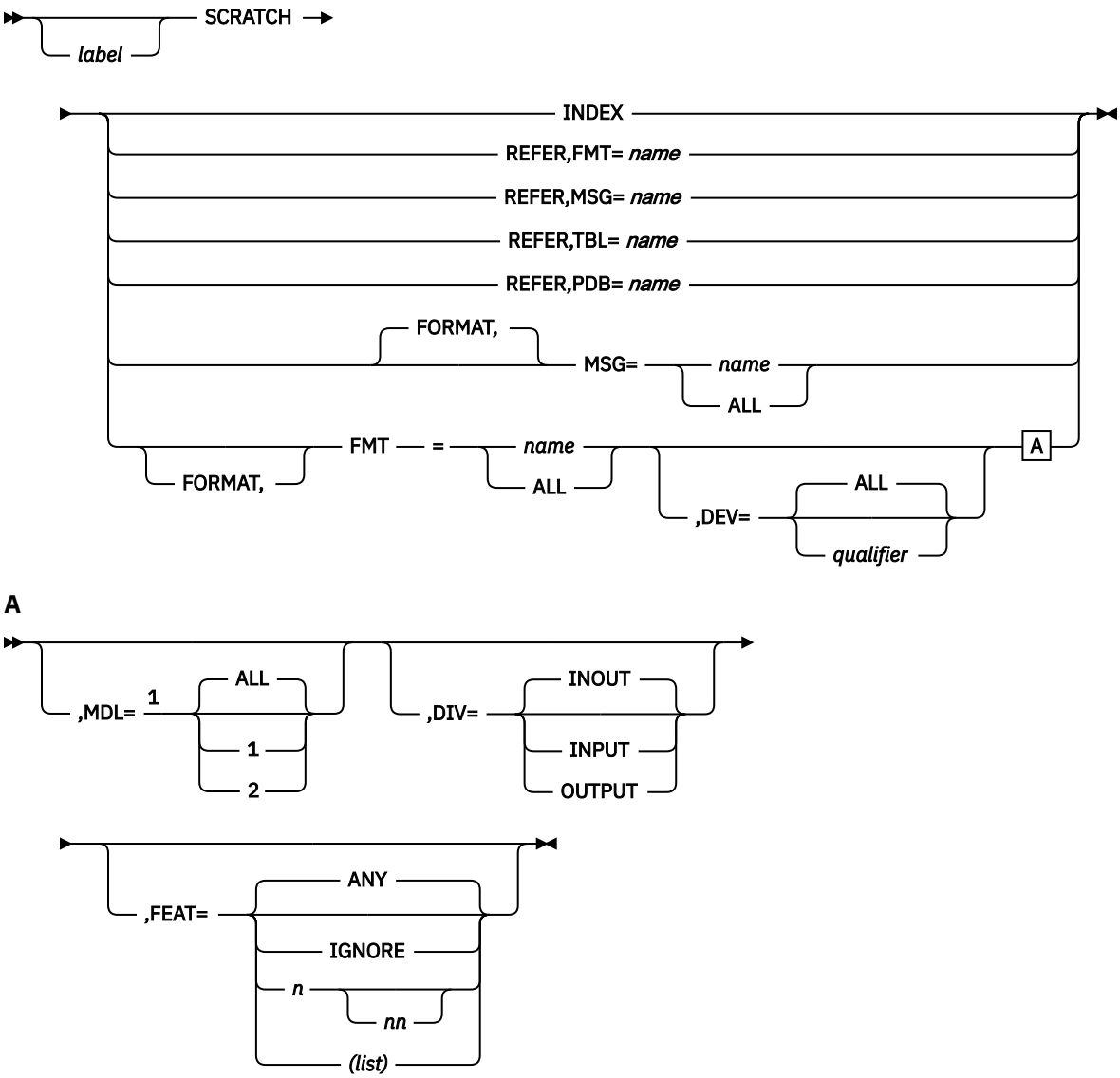


Notes:

[1] MDL=only applies to 3270 and 3270P devices.

SCRATCH function description

The SCRATCH function scratches a message, format, partition set, or table ITB from IMS.REFERAL. Using this function, you can also scratch a message descriptor, device format, or an index directory from IMS.FORMAT or IMS.TFORMAT.



Notes:

[1] MDL=only applies to 3270 and 3270P devices.

Effective use of the SCRATCH function requires an understanding of the relationship between the ITBs in IMS.REFERAL and the control blocks in IMS.FORMAT or IMS.TFORMAT. As shown in the following figure, a format set consists of a format and all associated messages where the SOR= parameter specifies the format as a source, and DFSDF2 is a format that is specified as the source for messages DFSMI2, DFSMO2, and DFSMO3. The ITB form of the format set is stored in IMS.REFERAL; the control block form is stored in IMS.FORMAT. The format ITB can correspond to a number of device formats for different device types and features.

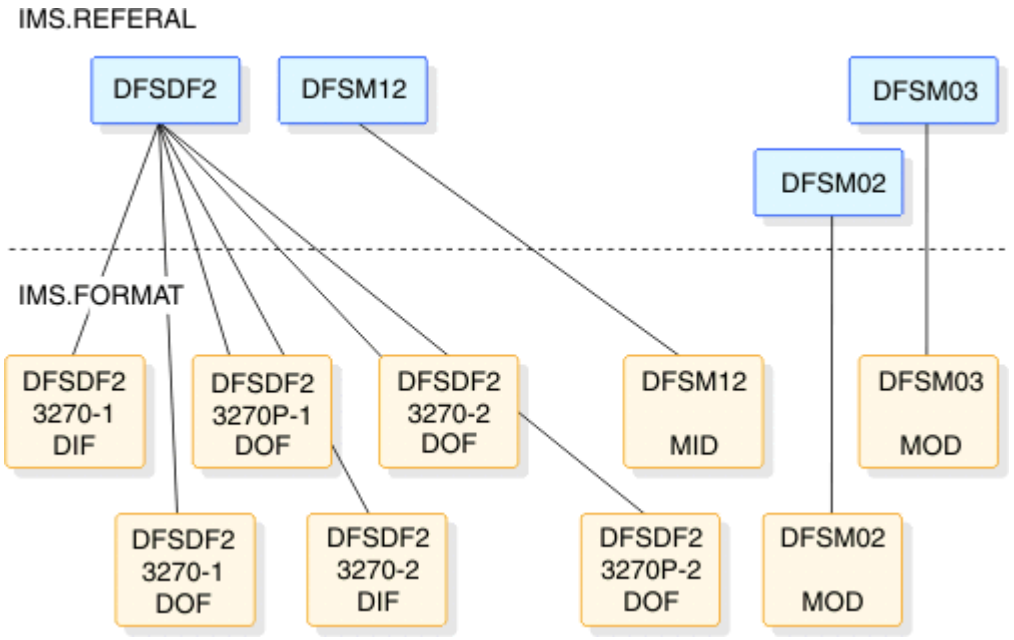


Figure 40. Relationships between ITBs in IMS.REFERAL and control blocks for 3270 format DFSDF2 and its format set

When a control block is scratched from IMS.FORMAT, the action is temporary; that is, the control block is restored to the staging library the next time any member of its format set is processed by the language utility. Thus, if the DFSDF2 DIF for 3270-1 were scratched from IMS.FORMAT, and DFSDF2 or one of its associated messages was later processed by the language utility, the DIF for 3270-1 would be placed in IMS.FORMAT again.

Total elimination of a control block requires the removal of its ITB as well. The 3270-1 control blocks for DFSDF2 could be deleted by recompiling the DFSDF2 format definition without the 3270-1 source included.

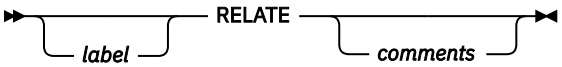
Another method is to use the MFS Service utility and scratch all of the DFSDF2 ITBs from the referral data set.

You can scratch the index directory using either DELETE ALL or SCRATCH INDEX.

Exception: The SCRATCH function does not apply to the MFS device characteristics table.

RELATE function description

The RELATE function provides a listing of all FMT, MSG, PDB, and TABLE ITBs in the IMS.REFERAL library.



FMT and MSG ITBs are related. Following each FMT ITB name, and indented three spaces, are the names of the MSG ITBs which include a SOR= parameter referencing that FMT. The MSG ITB entries indicate whether the message is INPUT or OUTPUT. A FMT ITB entry that does not exist in IMS.REFERAL, but is referred to by one or more MSG ITBs, is designated as **NOT DEFINED to the right of the entry. The PDB and TABLE ITBs are listed following the FMT and MSG ITBs.

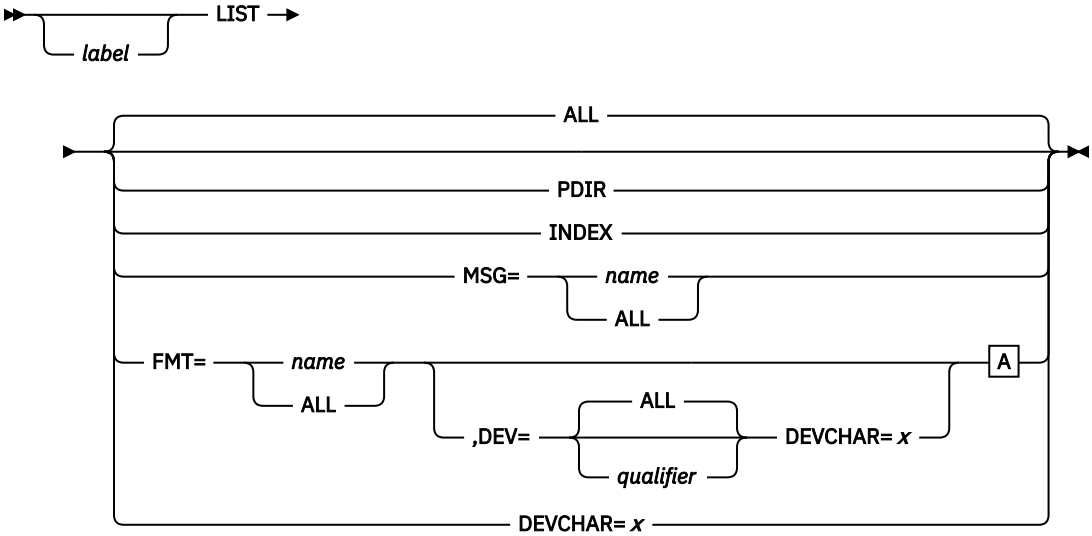
The following figure shows the contents of IMS.REFERAL and the relationships between the FMT and MSG ITBs. For example, the first entry shows DFSDF1 as a FMT name which has two MSGs associated with it: DFSMI1 (an input MSG) and DFSMO1 (an output MSG).

```
         RELATE
      DFSDF1
         DFSMI1        INPUT
```

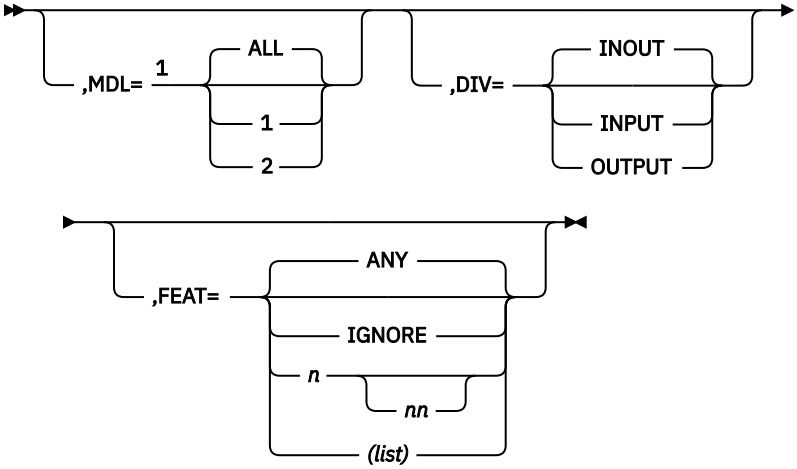
```
         DFSMO1        OUTPUT
      DFSDF2
         DFSDSPO1      OUTPUT
         DFSMI2        INPUT
         DFSMO2        OUTPUT

         DFSMO3        OUTPUT
         DFSM05        OUTPUT
      DFSDF3
         DFSMI4        INPUT
         DFSMO4        OUTPUT
      DSFDF5
         DFSMSTRI      INPUT
             DFS1209I  PROCESSING TERMINATED BY EOD ON SYSIN
```

LIST function description

The LIST function provides an interpreted listing of contents of the test library IMS.TFORMAT, the staging library IMS.FORMAT, or the device characteristics table DFSUDT0X.



A



Notes:

[1] MDL=only applies to 3270 and 3270P devices.

Function descriptions parameters

For the staging library and the test library you can select the contents to be listed by qualifying the INDEX, DELETE, SCRATCH, RELATE, and LIST control statement as follows:

ALL
Specifies both the PDS directory and the index directory. The default is ALL.

PDIR
Specifies the PDS directory.

INDEX
Specifies the index directory, $$IMSDIR.

MSG
Specifies the PDS directory entries for one or all message descriptors.

FMT
Specifies the PDS directory entries for one or more device formats.

A special case arises when the following chain of events occurs:

1. The Service Utility is started when the Index Directory ($$IMSDIR) does not exist.
2. The Index Directory is then created by the INDEX function.
3. LIST PDIR or LIST ALL is invoked.

In this case, the output containing the contents of the PDS directory does not include an entry for the Index Directory. This case occurs because the Index Directory is maintained in storage and is not written to the format library until the Service Utility ends.

The output listing contains a line for each control block with the following fields:

NAME
Represents the name of control block specified in the MSG or FMT statement.

TYPE
Represents the type of control block:

DIF
Device input format

DOF
Device output format

MSG
Message input or output control block

DEV
For FMT control blocks, represents the device type to which the control block applies.

MDL
For FMT control blocks, represents the model of the device type to which the control block applies. This field is used with 3270 and 3270P devices.

TTR
Represents the location of the control block in IMS.FORMAT (hexadecimal). Valid only for entries from IMS.FORMAT. Not valid for LIST INDEX.

SIZE
Represents the size of the control block in bytes (hexadecimal).

FEAT
For FMT control blocks, represents the hexadecimal uninterpreted representation of the device type, model, and specific features for which the control block applies. Because this information is appended to the FMT name in the library, this number determines the collating sequence of the output listing.

FEATURES
For FMT control blocks, represents the specific device features for which the control block applies.

The following figure shows an example of the LIST function output listing for the control blocks described in Figure 40 on page 556.

```
NAME      TYPE   DEV    MDL TTR        SIZE       FEAT   INTERPRETED FEATURES
-------------------------------------------------------------------------------
DFSDF2    DIF    3270    1  000705     000001AE   007F   IGNORE
DFSDF2    DOF    3270    1  000703     00000279   007F   IGNORE
DFSDF2    DOF    3270P   1  00070B     000001A5   017F   IGNORE
DFSDF2    DIF    3270    2  000709     000000B6   027F   IGNORE
DFSDF2    DOF    3270    2  000707     00000296   027F   IGNORE
DFSDF2    DOF    3270P   2  00070D     00000195   037F   IGNORE
DFSMI2    MSG               00070F     0000005E
DFSMO2    MSG               000711     00000006
DFSMO3    MSG               000713     000000F3
```

LIST the MFS device characteristics table

For the device characteristics table DFSUDT0x created during IMS system definition, the LIST DEVCHAR or DEVCHAR=0 or x, where x is the table suffix, provides an interpreted listing of all the entries of the MFS device characteristics table indicated. The listing shows the device symbolic name of the 3270 or SLU 2 devices, screen size, and features for each entry in the table. If the suffix is not specified in either the DEVCHAR parameter on the LIST statement or the PARM operand of the EXEC statement, table DFSUDT00 is listed. If the suffix is not specified in the DEVCHAR parameter or the LIST statement, but is specified in the EXEC statement, table DFSUDT0x (where x is the table suffix specified in the EXEC statement) is listed. The examples in Figure 41 on page 559 and Figure 42 on page 559 are provided for the LIST DEVCHAR function.

Figure 41. Example of a LIST DEVCHAR function with DEVCHAR=

```
//MFSRVC  EXEC  PGM=DFSUTSA0, PARM='DEVCHAR=3'
//SYSIN   DD     *
          LIST  DEVCHAR=3
          LIST  DEVCHAR=7
```

The first LIST statement lists the contents of the device characteristics table DFSUDT03.

The second LIST statement lists the contents of the device characteristics table DFSUDT07.

Figure 42. Example of a LIST DEVCHAR function

```
//MFSRVC   EXEC   PGM=DFSUTSA0
//SYSIN    DD     *
           LIST   DEVCHAR=0
           LIST   DEVCHAR=7
```

The first LIST statement lists the contents of the device characteristics table, DFSUDT00.

The second LIST statement lists the contents of the device characteristics table, DFSUDT07.

LIST DEVCHAR output

The output listing contains a line for each entry in the specified device characteristics table, which provides the following fields:

Symbolic Name
Symbolic name defined for the 3270 display in the TYPE= operand of the IMS system definition TYPE or TERMINAL macro.

Screen Size
Screen lines and columns of the 3270 display defined in the SIZE= operand of the IMS system definition TYPE or TERMINAL macro.

Device Features
Features specified in the FEAT= operand of the IMS system definition TYPE or TERMINAL macro.

The following figure is an example of a LIST DEVCHAR output.

```
SYMBOLIC            SCREEN SIZE           DEVICE
  NAME              LINES  COLS          FEATURES
-------------------------------------------------------------
3270-A01             12     80            CARD,PFK,PEN
3270-A02             24     80            IGNORE
3270-A03             32     80            CARD,DEKYBD,PEN
```

Return codes

The error message ERR TYPE x IN REFERAL LIBRARY, FUNCT=RELATE is issued if a problem exists in the MFS REFERAL library. The error types are:

1
Directory block length error

2
User appendage length error

3
Unknown block error

4
Duplicate DUMMY FORMAT error

5
Duplicate FORMAT error

6
TABLE block error

7
Message input block error

8
REFIN OPEN error

9
REFIN OPEN SYNAD taken

A
PDB block error

B
FORMAT block error

C
Message output block error

If an error is detected in the SYSIN, SYSPRINT, or REFIN <REFERAL> file, a return code is set to 4, 8 or 12 respectively.

If an error is detected on the RELATE function, the subsequent functions are ignored.

Chapter 29. Multiple Systems Verification utility (DFSUMSV0)

Use the Multiple Systems Verification utility (DFSUMSV0) to verify the consistency and compatibility of system definitions for IMS systems in a multisystem environment. Use this utility with IMS Multiple Systems Coupling (MSC) for all MSC link types.

This utility identifies errors that can prevent IMS systems from performing properly.

If you do not use this utility, you must manually verify the compatibility of system definitions.

Subsections:

- "Restrictions" on page 561
- "Prerequisites" on page 561
- "Requirements" on page 562
- "Recommendations" on page 562
- "Input" on page 562
- "Output" on page 564
- "JCL specifications" on page 569
- "Utility control statements" on page 570
- "Return codes" on page 571

Restrictions

The following restrictions apply to the Multiple Systems Verification utility:

- It cannot detect errors associated with Intersystem Communication (ISC).
- It does not support CICS.
- It cannot detect errors associated with directed routing.
- From 2 to 936 IMS systems can be verified in any one execution of the verification utility.
- It cannot be used to verify IMS subsystems within a shared queues group. MSC links between IMS subsystems in the same shared queues group are not supported and will cause message DFS2149 - RESTART ABORTED.
- If the remote logical terminals (LTERMs) are incorrectly defined, the utility recognizes and issues error messages only for the remote LTERMs, leaving the local LTERMs in the target area undetected.
- Only MSC descriptors associated with the MSC links defined within the IMS system being initialized are processed by IMS. All other MSC descriptors are ignored.
- When verifying IMS systems with different release levels, use the utility from the latest release level.
- DFSUMSV0 cannot verify resources that are defined by dynamic resource definition (DRD). Alternately, the /MSVERIFY command can be used to verify these resources.

Prerequisites

The following prerequisites apply to the Multiple Systems Verification utility.

Before executing this utility, you must resolve all IMS definition errors in all the multisystem control blocks to be included in the total multisystem configuration. After all system definitions are complete, you must bind all the multisystem control blocks from all the IMS multisystem definitions into IMS.SDFSRESL or some other user-specified library. The utility then has access to the multisystem control blocks.

© Copyright IBM Corp. 1974, 2024

For problem determination, assembly listings of all the IMS multisystem control blocks should be available when the utility is executed. Without the assembly listings, it will be difficult for you to resolve inconsistencies and incompatibilities displayed as a result of the multisystem verification process.

Before the verification utility can be executed, the multisystem control block modules for all systems to be verified must be loaded into IMS.SDFSRESL or some other user-specified library. The following figure provides a sample job stream that can load the multisystem control block modules into IMS.SDFSRESL.

```
//STEP1   EXEC PGM=IEWL,PARM='SIZE=(500K),NCAL,LET,REUS,XREF,LIST'
//SYSPRINT  DD SYSOUT=A
//SYSLMOD   DD DSNAME=IMS.SDFSRESL,DISP=OLD
//SYSUT1    DD UNIT=SYSDA,SPACE=(1024,(100,10))
//SYSLIN    DD *
  PLACE OBJECT MODULES HERE
  NAME DFSMSxxx(R)
/*
```

You must provide two or more object modules. A NAME statement identifying the preceding multisystem control block module (replace *xxx* with the 3-digit control block name suffix) must follow each object deck. If a library other than IMS.SDFSRESL is used, modify the //SYSLMOD statement accordingly.

Requirements

The following requirements apply to this utility.

The verification utility requires as input one or more control statements within a SYSIN data set.

Recommendations

The DFSUMSV0 utility has several recommendations related to its use and execution.

The Multiple Systems Verification utility should be run before attempting online executions to verify all defined links and routing paths.

If the MSVERIFY parameter is specified with the SYSTEM keyword on the IMSCTRL macro, only the IMS multisystem control block and verification utility are generated.

Input

This utility processes input in two phases:

- Input validation
- Multisystem control block verification

Input validation

After the input (control statements within a SYSIN data set) has been validated, a list of the valid multisystem control block names is printed. The utility then determines if the multisystem control block names are in the IMS.SDFSRESL PDS directory. The utility prints any control block names that are not found in the directory. If any errors are detected, the utility terminates execution with a completion code of 12. If no errors are found, the utility loads the multisystem control blocks into real storage.

Multisystem control block verification

The utility verifies the following specific portions of each multisystem control block:

- Partner IDs and assigned physical links
- Remote SYSID to Local SYSID Paths

- Remote and local transaction attributes
- Presence of corresponding logical terminals

Partner IDs and assigned physical links

The partner IDs in the logical link definitions are verified to ensure that a partner ID is:

- Not referenced in only one system
- Referenced in only two systems

Each partner ID, as defined with the PARTNER keyword on the MSLINK macro, is checked against every other partner ID in every other multisystem control block. Appropriate messages are printed if any errors are found. Logical links in error are treated as undefined in subsequent steps of the verification process.

When a partner ID is verified and there is also an MSPLINK (physical link) defined for this logical link in both systems, the physical link attributes are verified for type and buffer size. The following are physical link types:

- Real storage-to-real storage
- Channel-to-channel
- Virtual Telecommunications Access Method
- Transmission Control Protocol/Internet Protocol

If any physical link incompatibility is found, the attributes of both physical link definitions are displayed to assist you in determining the error. If the MSPLINK is defined for only one logical link, an information message is printed, indicating that the other end is undefined. In addition to the MSC physical and logical links that are defined to IMS through system definitions, you can identify remote names to IMS through an MSC descriptor. The MSC descriptor relates each remote resource with the link name of a generated MSNAME macro.

Remote SYSID to local SYSID paths

The SYSID table entries for a SYSID are verified across all multisystem control blocks for that SYSID number to determine if any path errors exist. A path error is an incomplete path between a multisystem control block in which the SYSID is defined as remote and the multisystem control block in which the SYSID is defined as local.

An incomplete path can occur for the following reasons:

- A SYSID in a multisystem control block for an intermediate system does not contain an address of an MSNAME block (undefined SYSID).
- The MSNAME block is assigned to a logical link block that has a path back to itself without a local SYSID (loop condition).
- The MSNAME block is associated with a logical link block that has an invalid partner ID. The partner ID is invalid if it is not defined in any other multisystem control block or if it is defined in more than one other multisystem control block.
- A SYSID number is defined as local in more than one system.
- A SYSID number is not defined as local in any system.

SYSIDs are scanned in numeric order until all logical link paths are verified. After a path is verified, the utility might display warning messages. These messages identify which logical link numbers this SYSID number-MSNAME should not be assigned to, because an invalid path to the local SYSID would result.

Remote and local transaction attributes

Each multisystem control block is scanned for remote transaction definitions. Each remote transaction definition references a remote and local SYSID. Each remote transaction code is compared with the

transaction codes in the system where the remote SYSID is defined as local. If no matching transaction code is found, an error message is printed. If a match is found, the attributes are verified in the two multisystem control blocks.

The following transaction code attributes must be consistent between systems:

- Local
- Remote
- Recoverable
- Nonrecoverable
- Conversational
- Fixed scratchpad area
- Fixed scratchpad area length
- Non-inquiry
- Inquiry
- Single segment
- Multisegment
- Non-Fast Path

If a discrepancy in the transaction attributes occurs, an error message is printed. The attributes specified for the transaction in both systems are displayed.

The return from the local transaction is checked to ensure a path exists out of the local system back to the corresponding remote transaction. If an error exists, a message is printed.

Presence of corresponding logical terminals

Each multisystem control block is scanned for remote LTERM definitions. Each LTERM is associated with an MSNAME block. Each MSNAME block contains remote and local SYSID definitions.

When a remote LTERM is found, the utility checks to ensure that a corresponding LTERM is defined with the same name in the system where the remote SYSID for the MSNAME is defined as local. If no corresponding LTERM definition is found, an error message is printed. If an LTERM is found, verification continues.

For multisystem LTERMs defined during IMS system definition, the remote LTERM definition can be made through the IMS system definition for the remote system or by the Extended Terminal Option (ETO) MSC descriptors for the remote system. When you use ETO MSC descriptors, the remote LTERMs do not exist until the initialization of the ETO feature on the remote IMS system. Therefore, an error message is printed by this utility for the missing remote LTERM indicating that the LTERM might be a dynamic resource.

The utility checks the return path from the destination LTERM to ensure that a path exists out of the local system back to the corresponding remote LTERM. If an error exists, a message is printed.

The return path to that system is the SYSID defined as local in the MSNAME block in the multisystem control block in which the LTERM was defined as remote.

After all verification work is complete, the utility prints a path map as an aid in visualizing the configuration of systems.

Output

The verification utility output includes information, warning, and error messages and a path map.

Because an error can cause other error conditions, messages with lower numbers should be analyzed and corrected first. The path map is a summary, in matrix format, of the routing paths verified by the utility. It is produced for the first 18 or fewer systems (in ascending sequence by multisystem control block name)

being verified. If more than 18 systems are being verified, verification for all systems occurs, appropriate messages for all systems are provided, but the path map is provided only for the first 18 systems.

The following figure shows the error messages and sample path map for a configuration of three systems with multisystem control block names of DFSMS001, DFSMS002, and DFSMS003. (For the sake of simplicity, these names are used to refer to the specific systems in the rest of this discussion.) The path map and error messages are extracted from a larger report, and not all error messages are shone.

The figure has three sections. The top section is a list of error messages and the middle and bottom sections show the path map. The middle section relates SYSIDs to specific systems. The bottom section relates partner IDs to logical links between specific systems.

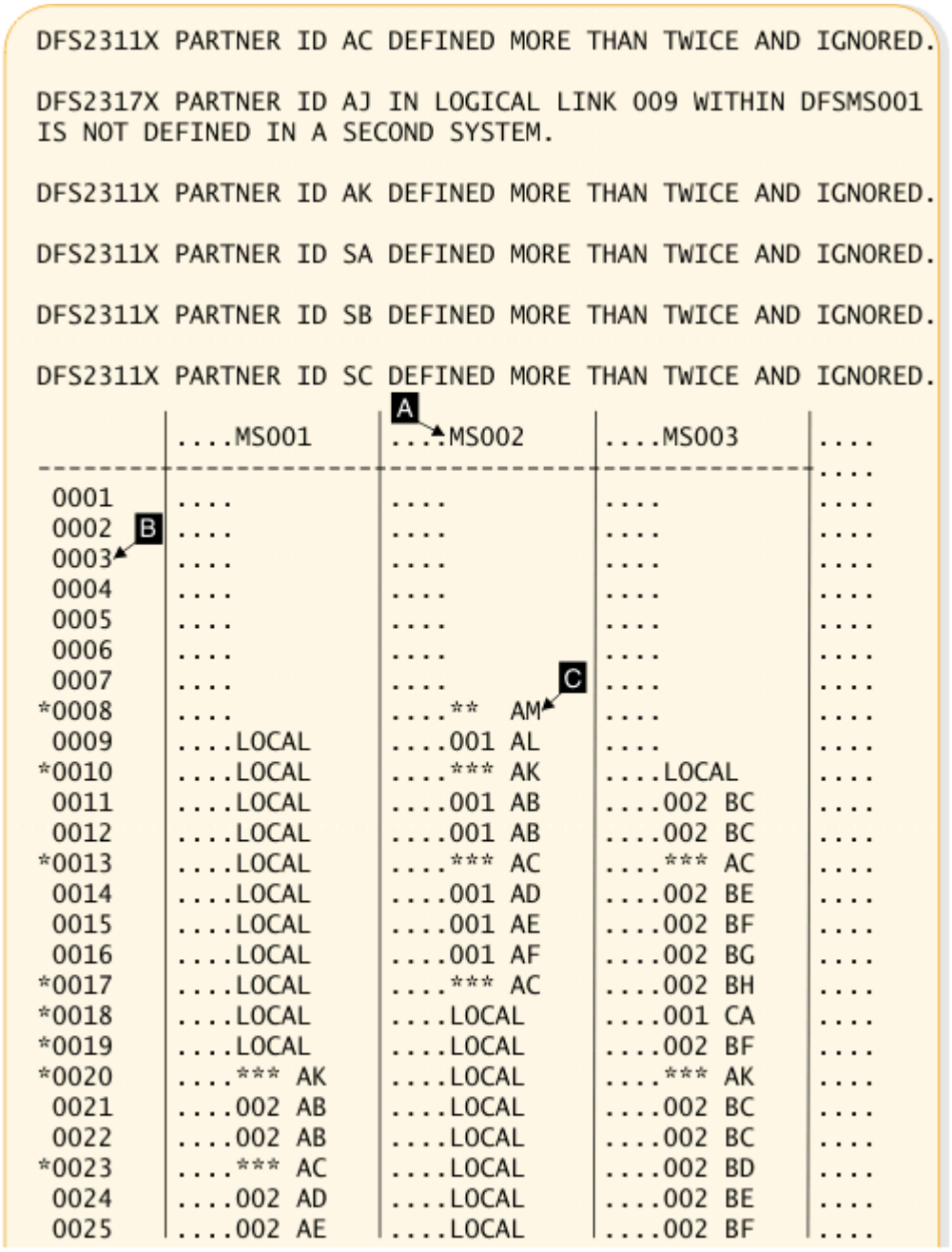


Figure 43. Sample error messages and multisystem path map

```
 0026 |....002 AF |....LOCAL |....002 BG |....
 0027 |....002 AG |....LOCAL |....002 BH |....
*0028 |....       |....001 AG |....       |....
*0029 |....002 AF |....001 AF |....LOCAL  |....
 0030 |....       |....       |....       |....
 0031 |....002 AB |....003 BC |....LOCAL  |....
 0032 |....002 AB |....003 BC |....LOCAL  |....
*0033 |....*** AC |....003 BD |....LOCAL  |....
 0034 |....002 AD |....003 BE |....LOCAL  |....
 0035 |....002 AE |....003 BF |....LOCAL  |....
*0036 |....*** AK |....003 BG |....LOCAL  |....
 0037 |....       |....003 BH |..         |
 0038 |....003 CA |....       |....LOCAL  |....
 0039 |....       |....       |....       |....
 0040 |....002 AL |....LOCAL  |....       |....
*0041 |....LOCAL  |....       |....002 BF |....
 0042 |....002 AH |....LOCAL  |....       |....
*0043 |....002 AH |....001 AH |....       |....
 0044 |....LOCAL  |....001 AH |....       |....
*0045 |....LOCAL  |....LOCAL  |....       |....
*0046 |....LOCAL  |....LOCAL  |....       |....
 0047 |....       |....       |....       |....
 0048 |....       |....       |....       |....
 0049 |....       |....       |....       |....
 0050 |....       |....       |....       |....
*0051 |....002 AG |....       |....LOCAL  |....
 0052 |....       |....       |....       |....
*0053 |....002 AI |....       |....       |....
*0054 |....**  AJ |....       |....       |....
 0055 |....       |....LOCAL  |....       |....
 0056 |....       |....       |....       |....
 0057 |....       |....       |....       |....
 0058 |....       |....       |....       |....
 0059 |....       |....       |....       |....
*0060 |....*** SA |....*** SA |....       |....
*0061 |....LOCAL  |....LOCAL  |....       |....
*0062 |....*** SB |....*** SB |....       |....
 0063 |....LOCAL  |....001 AI |....       |....
*0064 |....*** SC |....*** SC |....       |....
*0065 |....LOCAL  |....LOCAL  |....       |....
 0066 |....       |....LOCAL  |....       |....
 0067 |....       |....       |....       |....
 0068 |....       |....       |....       |....
 0069 |....       |....       |....       |....
```

```
  0070   |....          |....          |....          |....
  0071   |....LOCAL     |....001 BW    |....          |....
  0072   |....LOCAL     |....001 BX    |....          |....
  0073   |....LOCAL     |....001 BY    |....          |....
  0074   |....LOCAL     |....001 TA    |....          |....
  0075   |....LOCAL     |....001 TB    |....          |....
  0076   |....LOCAL     |....          |....001 TC    |....
  0077   |....LOCAL     |....001 XA    |....          |....
  0078   |....LOCAL     |....001 XB    |....          |....
  0079   |....LOCAL     |....          |....001 XC    |....
 *0080   |....*** AK    |....003 BF    |....LOCAL     |....
  0081   |....          |....          |....          |....
  0082   |....          |....          |....LOCAL     |....
  0083   |....          |....          |....          |....
  0084   |....002 TA    |....LOCAL     |....          |....
  0085   |....002 TB    |....LOCAL     |....          |....
  0086   |....          |....LOCAL     |....002 TD    |....
  0087   |....002 XA    |....LOCAL     |....          |....
  0088   |....002 XB    |....LOCAL     |....          |....
  0089   |....          |....LOCAL     |....002 XD    |....
 *0090   |....*** AK    |....LOCAL     |....          |....
  0091   |....002 BW    |....LOCAL     |....          |....
  0092   |....002 BX    |....LOCAL     |....          |....
  0093   |....002 BY    |....LOCAL     |....          |....
  0094   |....          |....LOCAL     |....002 BP    |....
  0095   |....          |....LOCAL     |....002 BQ    |....
  0096   |....          |....LOCAL     |....002 BR    |....
  0097   |....002 TB    |....LOCAL     |....001 TC    |....
  0098   |....          |....          |....          |....
  0099   |....          |....          |....          |....
  0100   |....LOCAL     |....          |....001 BS    |....
  0101   |....LOCAL     |....          |....001 BT    |....
  0102   |....LOCAL     |....          |....001 BU    |....
  0103   |....LOCAL     |....001 AN    |....          |....
  0104   |....002 AN    |....LOCAL     |....          |....
  0105   |....          |....          |....          |....
  0106   |....          |....          |....          |....
  0107   |....          |....          |....          |....
  0108   |....          |....          |....          |....
  0109   |....          |....          |....          |....
  0110   |....003 BS    |....          |....LOCAL     |....
  0111   |....003 BT    |....          |....LOCAL     |....
  0112   |....003 BU    |....          |....LOCAL     |....
```

```
 0113    |....          |....003 BP    |....LOCAL     |....
 0114    |....          |....003 BQ    |....LOCAL     |....
 0115    |....          |....003 BR    |....LOCAL     |....
 0116    |....003 TC    |....          |....LOCAL     |....
 0117    |....          |....003 TD    |....LOCAL     |....
 0118    |....003 XC    |....          |....LOCAL     |....
 0119    |....          |....003 XD    |....LOCAL     |....
 0120    |....003 TC    |....001 TB    |....LOCAL     |....
*2036    |....LOCAL     |....003 BF    |....002 BG    |....
---------------------------------------------------------....
 VTAM     ...AB.001--------AB.001.     .........  E  .......
* CTC   D ...AC.002--------AC.002--------AC.002*    .......
  MTM     ...AD.003--------AD.003.     .........    .......
          ...AE.004--------AE.007.     .........    .......
          ...AF.005--------AF.009.     .........    .......
          ...AG.006--------AG.010.     .........    .......
          ...AH.007--------AH.011.     .........    .......
          ...AI.008--------AI.012.     .........   . ......
* MTM      ..AJ.009.     .........     .........   . ......
*VTAM     ...AK.010--------AK.013--------AK.011.     .......
 VTAM     ...AL.011--------AL.014.     .........    .......
 VTAM     ...AN.028--------AN.033.     .........     ......
  MTM     ...BS.017----------------------BS.013.    .......
  CTC     ...BT.019----------------------BT.015.    .......
 VTAM     ...BU.021----------------------BU.017.    .......
  MTM     ...BW.016--------BW.021.     .........    .......
  CTC     ...BX.018--------BX.023.     .........    .......
 VTAM     ...BY.020--------BY.025.     .........    .......
 VTAM     ...CA.012----------------------CA.007.    .......
*VTAM     ...SA.013--------SA.018--------SA.008.    .......
*VTAM     ...SB.014--------SB.019--------SB.009.    .......
*VTAM     ...SC.015--------SC.020--------SC.010.    .......
  TCP     ...TA.022--------TA.027.     .........    .......
  TCP     ...TB.023--------TB.028.     .........    .......
  TCP     ...TC.024----------------------TC.019.    .......
---------------------------------------------------------.......
      *   = AN ERROR WAS DETECTED ON THIS LINE.
      **  = NO PARTNER FOR ID.
      *** = MULTI-PARTNERS FOR ID.

DFS2399I JOB TERMINATED - RETURN CODE 12
```

Notes for the figure:

Letter
Meaning

A
The top row contains the multisystem control block names (not including the DFS prefix) of the systems verified by execution of this utility.

B

The first column of the middle section of the figure contains all SYSIDs defined in the multisystem configuration. An asterisk preceding the SYSID number indicates an error exists on this line of the matrix.

C

Each entry in the middle section of the figure relates the SYSID (column B) to the multisystem control block name (row A).

- Most entries contain the 4-digit suffix of the multisystem control block name of the system defined as logically linked to the system (row A) and the 2-character partner ID defined for this logical link.
- A blank entry, such as SYSID 0041 for DFSMS002, indicates this SYSID was not defined for this system. All entries for SYSIDs 0001 through 0007 are blank; this means these SYSIDs were not specified in any of the multisystem control blocks.
- An entry specifying LOCAL, such as SYSID 0009 for DFSMS001, identifies this system as the system in which this SYSID is defined as local.
- Errors are identified by asterisks. One asterisk preceding the SYSID indicates that one or more errors were found for this printed line. If the suffix portion of an entry is replaced by two asterisks (**), such as SYSID 0008 for DFSMS002, the verification utility found no partner for this system. Three asterisks (***) indicate that more than two partners were found, such as SYSID 0017 for DFSMS002. If a SYSID is defined as local for more than one system, the printed line is identified by one asterisk and more than one entry on that line specifies LOCAL. SYSID 0010 contains an example of this error.

D

The first column of the bottom section contains the physical link type:

CTC
Channel-to-channel adapter

MTM
Real storage-to-real storage

TCP
Transmission Control Protocol/Internet Protocol (TCP/IP)

VTAM®
Virtual Telecommunications Access Method

This column entry is either blank, if no physical link is defined, or assigned depending on the first physical link encountered, for the logical link identified by E. An asterisk preceding the link type (or alone, if no physical link is defined) indicates an error exists for this printed line.

E

Identifies the logical link in terms of partner ID and relative logical link number. The partner IDs relate directly to those in the top part of the chart.

The verification utility relates partner systems by connecting them with a dashed line.

In the figure, partnerships AC, AJ, AK, SA, SB and SC are in error. These errors were identified in the top part of the chart but are more clearly demonstrated in the bottom part. Partner ID BC has more than two definitions; AX has just one definition.

JCL specifications

The DFSUMSV0 utility requires the procedure statement, the EXEC statement, and invoking the procedure.

The following figure shows an example of the procedure used to execute the Multiple System Verification utility. IMSMSV is created during SMP/E processing and is placed in the IMS.SDFSPROC.

```
//         PROC    DSN='IMS.SDFSRESL',
//                 REG=32K,CLASS=A,PARM='ALL',
```

```
//                 UNIT=SYSDA,SER=,DSM='IMS.MODBLKS'
//MSVERIFY EXEC    PGM=DFSUMSV0,PARM='&ALL',REGION=&REG
//STEPLIB  DD      DSN=&DSM,DISP=SHR,UNIT=&UNIT,VOL=SER=&SER
           DD      DSN=&DSN,DISP=SHR,UNIT=&UNIT,VOL=SER=&SER
//SYSOUT   DD      SYSOUT=&CLASS
```

Procedure statement

The procedure statement must be in the form:

```
PROC    DSN='IMS.SDFSRESL',
        REG=32K,CLASS=A,ALL=ALL,
        UNIT=SYSDA,SER=,DSM='IMS.MODBLKS'
```

ALL=
Specifies that all messages, including the information message DFS2327I, are to be printed. The default is ALL.

CLASS=
Specifies the SYSOUT class. The default is A.

DSM=
Specifies the data set name of IMS.MODBLKS.

DSN=
Specifies the data set name that contains the verification utility program (DFSMSV00) and its control blocks. The default is IMS.SDFSRESL.

REG=
Specifies the region size for this execution. The default is 32 KB.

SER=
Specifies the volume serial number of the DASD that contains the data set specified by the DSN parameter. SER= need not be specified if the data set is a cataloged data set.

UNIT=
Specifies the STEPLIB UNIT TYPE.

EXEC statement

The execution statement must be in the form of PGM=DFSUMSV0. The PARM= field must be in the form:

```
        PARM='&ALL',REGION=&REG
```

If PARM=ALL is specified in the EXEC statement for the utility execution, the information message DFS2327I is printed as part of the utility output. This message warns you not to do an //ASSIGN of the SYSID/MSNAME to the logical link referenced, because the assignment does not provide a path to the local SYSID at this SYSID level.

The following figure shows the JCL required to execute the DFSUMSV0 utility.

```
//STEP1   EXEC IMSMSV
//SYSIN     DD *
     INPUT FOR IMS MULTISYSTEM VERIFICATION UTILITY
/*
```

Utility control statements

The control statements must contain the 1- to 3-digit suffix supplied on the MSVID keyword of the IMSCTRL macro. Each control statement can contain one or more such suffixes, specified in any sequence. The input statement scan ends when a blank position is encountered; if position 1 is blank,

the input statement is treated as a comment statement. If more than one suffix is specified in a control statement, each suffix following the first one must be separated from the preceding suffix by a comma. Only the significant digits of a suffix need be specified.

Each suffix in a control statement must be complete in that statement and cannot be continued in the next control statement.

Sample input data:

- 1,255,6,009,80,02,198 are valid.
- 0,677,0040,NYC,1A,0001,5,5 are invalid.

Each invalid entry is printed, and the type of error is identified.

For example:

0
Is not in the range from 1 to 676.

677
Is not in the range from 1 to 676.

0040
Is more than 3 digits.

NYC
Is nonnumeric.

5,5
Is duplicate input data.

Valid input for three systems:

```
STATEMENT    1,5,255
   or
STATEMENT    001,005,255
   or
STATEMENT    255,01,5
```

Return codes

Code
Meaning

0
Only information and warning messages are printed

12
Errors detected that must be resolved before multisystem execution

Related reference
IMSCTRL macro (System Definition)
Related information
DFS2149 (Messages and Codes)

Chapter 30. Online Change Copy utility (DFSUOCU0)

Use the Online Change Copy utility (DFSUOCU0) as one step in the process of preparing an IMS or an IMSplex for a local or global online change.

The Online Change Copy utility copies a source library with your new definitions to a target library. Issuing the Online Change command sequence to prepare and commit an online change causes the inactive library to become the active library. Using a z/OS serialization service, this utility prevents other utilities from updating the staging (source) or inactive (target) libraries while the copy is in progress. The Online Change Copy Utility clears the target library and then invokes IEBCOPY to move the source library contents.

The Online Change Copy utility can copy the contents of the staging libraries to the active libraries during the installation of IMS, prior to the first cold start. To do this, the parameter for the output ddname must be specified when invoking the utility, because the initial contents of IMS.MODSTAT (or OLCSTAT data set, if global online change is enabled) will specify the active libraries.

The Online Change Copy utility (DFSUOCU0) can be used to create a backup copy of the active ACB library, which can be used for fallback purposes. This backup copy is created by specifying all of the following elements:

- The copy type parameter, TYPE=ACTVACB, which specifies that the Online Change Copy utility copies the members from the active ACB library

  Attention: The OLCSTAT library must be initiated before running the Online Change Copy utility with TYPE=ACTVACB specified.
- The target library parameter, OUT=O, which specifies that the Online Change Copy utility copies the content of an active ACB library to a library other than the inactive ACB library
- The IMSACBO DD statement, which specifies where the backup copy is created

In an IMSplex where IMS subsystems are not cloned and the libraries not shared, the Online Change Copy utility might need to be executed on every IMS in the IMSplex. In an IMSplex where IMS subsystems are cloned and the libraries are shared, the Online Change Copy utility might need to be executed only once on one IMS.SDFSRESL at the highest IMS level.

Subsections:

- “Restrictions” on page 573
- “Prerequisites” on page 574
- “Requirements” on page 574
- “Recommendations” on page 574
- “JCL specifications” on page 574

Restrictions

The following restrictions apply to the Online Change Copy utility:

- If any of the ACBLIB, FORMAT, or MODBLKS libraries are shared among IMS systems, all systems must use the same libraries during execution of this utility.
- In an XRF environment ACBLIBs, FORMATs, and MODBLKs data sets must be on shared non-duplexed DASD for error protection. Make the same additions or changes to separate but duplicate copies of IMS data sets.
- You cannot use the Online Change Copy utility to make additions or changes that require new IMS modules to be added to the IMS.SDFSRESL data set.

© Copyright IBM Corp. 1974, 2024

- You cannot add, modify, or delete MSDBs using this utility. However, PSB related changes for MSDBs can be made to the ACBLIBs, as long as no DBD changes are included.
- You cannot add, modify, or delete partitions of a HALDB database using this utility; only by using the Partition Definition utility. However, PSB related changes for HALDB database can be made to the ACBLIBs, as long as no DBD changes are included.
- If dynamic resource definition (DRD) is enabled (MODBLKS=DYN is specified in the DFSCGxxx IMS.PROCLIB member or in the COMMON_SERVICE_LAYER section of the DFSDFxxx IMS.PROCLIB member), the Online Change Copy utility might not be needed. Because the online change function is disabled, you do not need to copy stored resource definitions to prepare for an online change. You can use the Online Change Copy utility to copy stored resource definitions to a MODBLKS data set that will be loaded by an IMS cold start.

If the Online Change Copy utility is canceled prior to completion, the status of the ACBLIB, FMTLIB, or MODBLKS data set is unpredictable. The data set that is being changed by the utility is cleared as soon as the utility has exclusive control of the data set, and then new information is written to the data set. If the utility is canceled prior to its successful completion the information in the data set is not usable.

Prerequisites

Currently, no prerequisites are documented for the DFSUOCU0 utility.

Requirements

Three copies of the following libraries are required for online change:

ACBLIB
Database and program descriptors such as DMBs and PSBs

FORMAT
Control blocks produced by the MFS Language utility and service utility

MODBLKS
A subset of the control blocks for the resources to be modified

One copy of each library is used exclusively for offline functions. This library has no suffix and is called the staging library.

The other two copies of each library have a suffix of A or B. Only one of these libraries is used by the IMS online system at any one time. The one in use is referred to as the active library. The other is called the inactive library.

The Online Change Copy utility can copy the contents of the staging library to the inactive library based on the information in the MODSTAT data set (local online change) or the OLCSTAT data set (global online change).

The same method of serialization prevents this utility from updating the active libraries while they are being used by an IMS online system.

Recommendations

Create a backup copy of all the active ACB library members before you perform an ACB library member online change operation.

JCL specifications

Procedure statement

The procedure statement must be in the following form to invoke the utility and to include optional IEBCOPY parameters:

```
OLCUTL PROC TYPE=,IN=,OUT=,SOUT=A,SYS=,SYS2=,OLCGLBL='DUMMY,',OLCLOCL=
```

OLCGLBL=
OLCLOCL=
These parameters produce the OLCSTAT DD card or the MODSTAT/MODSTAT2 DD cards.

OLCGLBL='DUMMY,',OLCLOCL=
Built during SMP/E processing and placed in the ADFSPROC and SDFSPROC data sets. This parameter results in the OLCUTL procedure being set up for local online change as the default with the following DD statements:

```
//MODSTAT  DD &OLCLOCL.DSN=IMS.&SYS.MODSTAT,DISP=SHR
//MODSTAT2 DD &OLCLOCL.DSN=IMS.&SYS.MODSTAT2,DISP=SHR
//OLCSTAT  DD &OLCGLBL.DSN=IMSPLEX.OLCSTAT,DISP=OLD
```

For global online change, set the OLCGLBL= and OLCLOCL= parameters as follows:

```
OLCGLBL=,OLCLOCL='DUMMY,'
```

These parameters generate the following DD statements to be used for global online change:

```
//MODSTAT DD DUMMY,DSN=IMS.&SYS..MODSTAT,DISP=SHR
//MODSTAT2 DD DUMMY,DSN=IMS.&SYS..MODSTAT2,DISP=SHR
//OLCSTAT DD DSN=IMSPLEX.OLCSTAT,DISP=OLD
```

SOUT=
Specifies the class assigned to SYSOUT DD statements.

SYS=
Specifies an optional second level dsname qualifier for those data sets which are designated as "Mandatory Shared" in an XRF complex. When specified, the parameter must be enclosed in quotes and must include a trailing period; for example, SYS='IMSA.'

SYS2=
Specifies an optional second level dsname qualifier for those data sets which are designated as "Optional Replicate" in an XRF complex. When specified, the parameter must be enclosed in quotes and must include a trailing period; for example, SYS2='IMSA.'

EXEC statement

The EXEC statement determines which copy is made and which data sets are used for input and output. The format of this statement can include optional IEBCOPY parameters specified in the order of WORK, SIZE, LIST after the target library.

When you omit a parameter, include a comma in its place. For example, when you specify only the LIST parameter without specifying WORK or SIZE, use the following syntax: `,,&LIST`

The IEBCOPY options list, which contains the IEBCOPY keywords, equal signs, parameter values, and commas, cannot exceed 64 bytes. The IEBCOPY parameters are passed to the IEBCOPY utility when the utility is called.

The EXEC statement must be in the following form:

```
PGM=DFSUOCU0,PARM=(&TYPE,&IN,&OUT,&WORK,&SIZE,&LIST)
```

&TYPE
Specifies the type of library to be copied from the input library to the target library. The copy type can be the ACB, ACTVACB, FORMAT, or MODBLKS library. ACTVACB is unique and different from the other three.

Parameter
Meaning

ACTVACB
Specifies that the active ACB library members are to be copied to create a backup fallback purposes. When the &TYPE parameter ACTVACB is specified, no input library parameter (&IN) is necessary or supported. If the &IN parameter is coded with the &TYPE parameter ACTVACB, IMS issues an informational message (DFS3469I).

&IN
Defines the library ddnames to be used as input.

Parameter
Meaning

S
IMS staging library (IMSACB, FORMAT, or MODBLKS)

I
User input library (IMSACBI, FORMATI, or MODBLKSI)

The I parameter allows you to use an input library other than the staging library.

&OUT
Defines the library ddnames to be used for output.

Parameter
Meaning

A
IMS A library (IMSACBA, FORMATA, or MODBLKSA)

B
IMS B library (IMSACBB, FORMATB, or MODBLKSB)

G
Target library (inactive) determined by the utility, using the OLCSTAT data set. The target is the library not currently in use by the IMS online system. The OLCSTAT data set can either be allocated by the OLCSTAT DD statement, or dynamically by specifying the OLCSTAT dataset name starting in column 1 of the SYSIN DD statement.

When G is specified with the ACTVACB parameter, the Online Change Copy utility copies the contents of the active ACB library to the inactive ACB library.

In this case, the OLCUTL JCL must include a DD statement, which specifies the data set name for the backup copy of the active ACB library.

O
User output library (IMSACBO, FORMATO, or MODBLKSO)

When O is specified with the ACTVACB parameter, the Online Change Copy utility copies the contents of the active ACB library to a library other than the inactive ACB library.

In this case, the OLCUTL procedure JCL must include a DD statement with a data definition name (ddname) of IMSACBO, which specifies the data set name for the backup copy of the active ACB library.

The O parameter allows you to select a target data set other than the active or inactive data set.

U
Target library (inactive) determined by the utility, using the MODSTAT data set. The target is the library not currently in use by the IMS online system.

Recommendations: During online operation, avoid using the A or B parameter for the output library because an incorrect choice could cause IMS to overlay the active library.

Specify the U parameter for an IMS that supports local online change. G is recommended for an IMSplex that supports global online change.

&WORK
Optional parameter that passes the work parameter to the IEBCOPY utility. The work parameter passes the number of bytes of virtual storage to request for a work area to hold for directory entries, internal tables, and I/O buffers.

&SIZE
Optional parameter that passes the size parameter to the IEBCOPY utility. The size parameter specifies the maximum number of bytes of virtual storage that the IEBCOPY utility can use as a buffer.

&LIST
Optional parameter that passes the list parameter to the IEBCOPY utility. LIST=NO suppresses IEBCOPY IEB1541 messages that are issued for each member that is successfully copied.

DD statements

IMSACB DD
IMSACBA DD
IMSACBB DD
Defines the staging, active, or inactive ACBLIB.

IMSACBI DD
User-defined input ACBLIB.

IMSACBO DD
User-defined output ACBLIB.

FORMAT DD
FORMATA DD
FORMATB DD
Defines the staging, active, or inactive MFS format library.

FORMATI DD
User-defined input FORMAT library.

FORMATO DD
User-defined output FORMAT library.

OLCSTAT DD
Defines the global online change status data set name for an IMS enabled for global online change. The OLCSTAT data set is similar to the MODSTAT data set used for local online change. The OLCSTAT DD should not be defined if local online change is enabled, or if global online change is enabled and the OLCSTAT data set is to be dynamically allocated.

The SYSIN DD statement may be specified for &OUT option G, in order to dynamically allocate the OLCSTAT data set, rather than define it with the OLCSTAT DD statement. The OLCSTAT data set must be defined as data after the SYSIN DD statement, starting in column 1.

If the OLCSTAT data set is defined with the OLCSTAT DD statement and the SYSIN DD statement, the SYSIN DD statement is ignored.

If a multi-step batch job invokes the DFSUOCU0 utility in a step after a CSLUSPOC step that issues INITIATE OLC commands, the DFSUOCU0 utility must allocate the OLCSTAT data set dynamically.

MODBLKS DD
MODBLKSA DD
MODBLKSB DD
Defines the staging, active, or inactive system definition library.

MODBLKSI DD
User-defined input MODBLKS library.

MODBLKSO DD
User-defined output MODBLKS library.

MODSTAT DD
MODSTAT2 DD
Defines the local online change modify status data set names. This is the active data set (and inactive data set, if XRF is used) that online IMS should use at initialization.

SYSUDUMP DD
Defines the dump data set for this program. The data set can reside on a printer, tape, or direct-access device, or be routed through the output stream.

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

SYSUT3 DD
Defines a work data set that is required.

SYSUT4 DD
Same function as SYSUT3.

COPYCTL DD
Defines a copy control data set to be built prior to calling IEBCOPY.

Related reference
z/OS: IEBCOPY (Library Copy) program
"Global Online Change utility (DFSUOLC0)" on page 541
Use the Global Online Change utility to initialize, recreate, or unlock the OLCSTAT data set.

OLCUTL procedure

Use the OLCUTL procedure to invoke the DFSUOCU0 utility, specifying the parameters you select for the library type, input library, output library, and IEBCOPY parameters.

Parameters

You can specify the following execution parameters in the OLCUTL procedure:

- NODE1
- NODE2
- OLCPLEX: specifies the data set qualifier or qualifiers for the OLCSTAT data set, which corresponds to the IMSPLEX parameter of the IMSGEN macro.

NODE1 and NODE2 are described in Parameter descriptions for IMS procedures (System Definition)

Invoking the OLCUTL procedure

The following figure shows the OLCUTL statements used to invoke the Online Change Copy utility, OLCUTL procedure.

Recommendation: OLCUTL clears the target library data set and then invokes IEBCOPY to move the source library contents. If IEBCOPY abends because of insufficient space, the contents of the target library are unpredictable. To avoid this, allocate equal amounts of space for source and target libraries.

```
//       PROC TYPE=,IN=,OUT=,SOUT=A,SYS=,SYS2=,OLCGLBL='DUMMY,',OLCLOCL=,
//               OLCPLEX=IMSPLEX,
//               NODE1=IMS,
//               NODE2=IMS
//S      EXEC PGM=DFSUOCU0,PARM=(&TYPE,&IN,&OUT)
//STEPLIB  DD DSN=&NODE2..&SYS2.SDFSRESL,DISP=SHR
//MODBLKS  DD DSN=&NODE2..&SYS2.MODBLKS,DISP=SHR
//MODBLKSA DD DSN=&NODE2..&SYS2.MODBLKSA,DISP=SHR
//MODBLKSB DD DSN=&NODE2..&SYS2.MODBLKSB,DISP=SHR
//IMSACB   DD DSN=&NODE1..&SYS2.ACBLIB,DISP=SHR
//IMSACBA  DD DSN=&NODE1..&SYS2.ACBLIBA,DISP=SHR
//IMSACBB  DD DSN=&NODE1..&SYS2.ACBLIBB,DISP=SHR
//FORMAT   DD DSN=&NODE1..&SYS2.FORMAT,DISP=SHR
//FORMATA  DD DSN=&NODE1..&SYS2.FORMATA,DISP=SHR
//FORMATB  DD DSN=&NODE1..&SYS2.FORMATB,DISP=SHR
```

```
//MODSTAT  DD &OLCLOCL.DSN=&NODE1..&SYS.MODSTAT,
//         DISP=SHR
//MODSTAT2 DD &OLCLOCL.DSN=&NODE1..&SYS.MODSTAT2,
//         DISP=SHR
//OLCSTAT  DD &OLCGLBL.DSN=&OLCPLEX..OLCSTAT,
//         DISP=OLD
//SYSUDUMP DD SYSOUT=&SOUT
//SYSPRINT DD SYSOUT=&SOUT
//SYSUT3   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//COPYCTL  DD DSN=&&COPYCTL,DISP=(NEW,DELETE),
//         UNIT=SYSDA,SPACE=(CYL,(1,1))
```

Copying the staging libraries

The following figure shows the JCL statements used to copy the staging libraries to the inactive libraries indicated by MODSTAT data set.

```
//*  COPY MODBLKS TO MODBLKSA
//*
//STEP01  EXEC PROC=OLCUTL,SOUT='*',TYPE=MODBLKS,IN=S,OUT=U,,,LIST=NO
//*
//*  COPY ACBLIB TO ACBLIBA
//*
//STEP03  EXEC PROC=OLCUTL,SOUT='*',TYPE=ACB,IN=S,OUT=U
//*
//*  COPY FORMAT TO FORMATA
//*
//STEP04  EXEC PROC=OLCUTL,SOUT='*',TYPE=FORMAT,IN=S,OUT=U
//*
//*  COPY FORMAT TO FORMAT WITH IEBCOPY PARAMETERS SPECIFIED
//*
//STEP05  EXEC PROC=OLCUTL,SOUT='*',TYPE=FORMAT,IN=S,OUT=U,WORK=2M,SIZE=2M,LIST=NO
//*
//*
```

Backing up an active ACB library for Member OLC

The following figure shows how the active ACB library is copied to the inactive ACB library (OUT=G).

```
//DFSUOCU0 JOB LINK,MSGLEVEL=1,REGION=640K,CLASS=N,
           USER=USRID01
//STEP1    EXEC OLCUTL,TYPE=ACTVACB,OUT=G,SOUT=*,
//         OLCLOCL='DUMMY,',OLCGLBL=,SYS=
//
```

The following figure shows how the active ACB library is copied to the ACB library that is specified on the IMSACBO DD statement in the OLCUTL procedure (OUT=O).

```
//DFSUOCU0 JOB LINK,MSGLEVEL=1,REGION=640K,CLASS=N,
           USER=USRID01
//STEP1    EXEC OLCUTL,TYPE=ACTVACB,OUT=O,SOUT=*,
//         OLCLOCL='DUMMY,',OLCGLBL=,SYS=
//
//
```

Initializing the IMS.MODSTAT data set

The INITMOD procedure initializes the IMS.MODSTAT data set, for an IMS enabled with local online change. The MODSTAT data set must be initialized before the first IMS cold start or before any other cold start if IMS.MODSTAT does not contain the current ddnames.

If IMS is enabled for global online change, it will not use the MODSTAT data set.

Recommendation: Do not define the MODSTAT DD cards if enabling global online change, and the MODSTAT data sets will not have to be defined.

The procedure is built during SMP/E processing and placed in the ADFSPROC and SDFSPROC data sets.

```
//INIT1  EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=&SOUT
//SYSUT2   DD DSN=&NODE1..&SYS.MODSTAT&SF,DISP=OLD
//SYSIN    DD DUMMY
//SYSUT1   DD DISP=SHR,
//         DSN=&NODE2..&SYS2.PROCLIB(DFSMREC)
```

Procedure statement

The procedure statement must be in the form:

```
PROC SYS=,SYS2=,SF=,SOUT=A,NODE1=IMS,NODE2=IMS
```

SYS=
Specifies an optional second level dsname qualifier for those data sets which are designated as "Mandatory Shared" in an XRF complex. When specified, the parameter must be enclosed in quotes and must include a trailing period; for example, SYS='IMSA.'

SYS2=
Specifies an optional second level dsname qualifier for those data sets which are designated as "Optional Replicate" in an XRF complex. When specified, the parameter must be enclosed in quotes and must include a trailing period; for example, SYS2='IMSA.'

SF=
Specifies the suffix for the MODSTAT data set name, either SF= or SF=2.

SOUT=
Specifies the class assigned to SYSOUT DD statements.

DFSMREC control statement

The INITMOD procedure uses this control statement to initialize the MODSTAT data sets. DFSMREC contains the data for the MODSTAT record. This control statement is created by using the DFSPROCB JCL that is built during SMP/E processing. The statement must be in the form:

```
0,MODBLKSA,IMSACBA,FORMATA
```

Values must be separated by commas, with no imbedded blanks.

0
Is the MODSTAT identifier, which is variable length with no limit. This positive value, initialized to zero, is used by IMS internal processing for recovery of security status during emergency restart processing. You can initialize it to zero at any IMS cold start.

MODBLKSA | MODBLKSB
Is the ddname for the active MODBLKS data set, either IMS.MODBLKSA or IMS.MODBLKSB data set that contains the IMS system definition output.

If DRD is enabled, you can start IMS without defining MODBLKS DD cards. The INITMOD procedure requires that you define either MODBLKSA or MODBLKSB to initialize the MODSTAT data set, even for an IMS that does not define the MODBLKS data sets. If the MODBLKS data sets are not defined to IMS, IMS ignores the MODBLKSA or MODBLKSB definition in the MODSTAT data set.

The INITMOD procedure requires that you define either MODBLKSA or MODBLKSB to initialize the MODSTAT data set, even for an IMS that does not define the MODBLKS data sets.

IMSACBA | IMSACBB
Is the ddname in the IMS procedure for the active IMSACB data set, either IMS.ACBLIBA or IMS.ACBLIBB library.

FORMATA | FORMATB

The ddname of the active FORMAT data set, either IMS.FORMATA or IMS.FORMATB data set contains online MFS definitions to be used as the format library by the online system. MFS-supported terminals and the MFS Language utility program require their use. When one of these libraries is active (that is, in use by the online system), the contents of IMS.FORMAT is copied to the other, or inactive, library for use in the next online change run. Their ddnames must be FORMATA and FORMATB, respectively. If MFS is not defined, IMS ignores this ddname.

For DBCTL, you can start a DBCTL IMS without defining FORMAT DD cards. The INITMOD procedure requires that you define either FORMATA or FORMATB to initialize the MODSTAT data set, even for a DBCTL IMS that does not use the FORMAT data sets. A DBCTL IMS ignores the FORMATA or FORMATB definition in the MODSTAT data set.

The INITMOD procedure requires that you define either FORMATA or FORMATB to initialize the MODSTAT data set, even for an IMS that does not define the FORMAT data sets.

If the IMS.MODSTAT record contents are lost and must be reconstructed, or if you do not use default initialization by the INITMOD procedure, you must run an IEBGENER job to construct its contents with the proper values for the online change identifier and ddnames. The attributes for a new IMS.MODSTAT data set should be RECFM=F and BLKSIZE=80.

The following figure shows initialization of the MODSTAT ID to 0, and the ddnames to MODBLKSA, IMSACBA, and FORMATA.

```
//INIT1  EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=&SOUT
//SYSUT2   DD DSN=IMS.&SYS.MODSTAT&SF,DISP=OLD
//SYSIN    DD DUMMY
//SYSUT1   DD DISP=SHR,
//         DSN=IMS.&SYS2.PROCLIB(DFSMREC)
./       ADD  NAME=DFSMREC
./       NUMBER NEW1=10,INCR=10
0,MODBLKSA,IMSACBA,FORMATA
```

Alternatively, you can override SYSUT1 and SYSUT2 DD statements of the INITMOD procedure to accomplish the same purpose as the preceding IEBGENER sample job.

The DFS3499 message, which identifies the current values of the MODSTAT record, follows the DFS994 checkpoint message. The DFS3410 message at initialization also identifies the MODSTAT record data.

Chapter 31. Spool SYSOUT Print utility (DFSUPRT0)

Use the Spool SYSOUT Print Utility (DFSUPRT0) to copy messages produced by the online control program from its set of data sets to a system output device when a communication line is defined for Spool SYSOUT during system definition.

Both the spool data sets and the system output device are processed using QSAM. Blocking factors for spool data sets are determined by the online control program. System output device blocking can be specified through JCL on the SYSPRINT DD statement.

Subsections:

- "Restrictions" on page 583
- "Prerequisites" on page 583
- "Requirements" on page 583
- "Recommendations" on page 583
- "Input and output" on page 583
- "JCL specifications" on page 584
- "Return codes" on page 585

Restrictions

The Spool SYSOUT Print utility does not support CICS.

Prerequisites

Currently, no prerequisites are documented for the DFSUPRT0 utility.

Requirements

Currently, no requirements are documented for the DFSUPRT0 utility.

Recommendations

After the print utility is started, let it complete before issuing a **/START LINE** command to make the Spool SYSOUT available.

Input and output

Output from the print utility includes a page of status information, followed by the contents of the spool data sets indicated as FULL and printed in chronological sequence.

Sample output

The following figure shows an example of the Spool SYSOUT Print utility output.

```
DFSUPRTO - SYSOUT PRINT UTILITY    TIME   9:35:1   DATE    10.056

DDNAME    STATUS   CREATED TIME     DATE     DATASET NAME
SPOOL1     FULL      9:33:239      10.244    IMSTESTL.IMS01.SPOOL1
SPOOL2     INUS       :00:000       0.000    IMSTESTL.IMS01.SPOOL2
SPOOL3     AVAL       :00:000       0.000    IMSTESTL.IMS01.SPOOL3
```

© Copyright IBM Corp. 1974, 2024

The fields in the report have the following meaning:

DDNAME
The user-provided DDNAME.

STATUS
FULL—if data set is to be printed.

INUS—if being filled online.

AVAL—if not being used.

CREATED TIME
Time (24-hour clock) (HH:MM:SST) when the data set was marked FULL.

DATE
Julian date (YY.DDD) when the data set was marked FULL.

DATASET NAME
The DSNAME of the assigned data set.

System messages included in a spool data set always have unprintable control characters (typically the new-line symbol, X'15').If a UCS printer is used as a SYSOUT device, these messages might print as extraneous alphabetic characters (if fold-mode operation is specified in response to the UCS parameter request).

JCL specifications

The DFSUPRT0 utility requires a procedure statement, the EXEC statement, and the DD statements.

The DFSWT000 procedure, shown in the following figure, executes the Spool SYSOUT Print utility program (DFSUPRT0) as an online program for printing data sets created by the Spool SYSOUT option during system definition. The utility program copies messages produced by the online control program from its set of data sets to a system output device. The procedure is built during SMP/E processing and placed in the ADFSPROC and SDFSPROC data sets.

```
//       PROC  SOUT=A,RGN=30K,SYS1=,SYS2=,
//                 NODE1=IMS,
//                 NODE2=IMS
//PRINT  EXEC  PGM=DFSUPRT0,REGION=&RGN
//STEPLIB  DD  DSN=&NODE2..&SYS2.SDFSRESL,DISP=SHR
//*
//SYSPRINT DD SYSOUT=&SOUT,DCB=BLKSIZE=1410
//SYSUDUMP DD SYSOUT=&SOUT
//*
//SPOOLn  DD  DISP=SHR,DSN=&NODE1..&SYS1.SYSO1
//SPOOLn  DD  DISP=SHR,DSN=&NODE1..&SYS1.SYSO2
//SPOOLn  DD  DISP=SHR,DSN=&NODE1..&SYS1.SYSO3
//*
```

Procedure statement

This statement must be in the form:

```
PROC  SOUT=A,RGN=30K,SYS1=,SYS2=
```

SOUT=
Specifies the class assigned to SYSOUT DD statements.

RGN=
Specifies the size of the z/OS region to be allocated to the IMS control program.

SYS1=
Specifies an optional second level dsname qualifier for those data sets which are designated as "Mandatory Replicate" in an XRF complex. When specified, the operand must be enclosed in quotes and must include a trailing period; for example, SYS1='IMSA.'.

SYS2=
Specifies an optional second level dsname qualifier for those data sets which are designated as "Optional Replicate" in an XRF complex. When specified, the operand must be enclosed in quotes and must include a trailing period; for example, SYS2='IMSA.'.

EXEC statement

This statement can be in the form:

```
EXEC   PGM=DFSUPRT0
```

Or it can specify a procedure that contains the required JCL statement. A region size of 30KB is usually adequate for execution.

DD statements

STEPLIB DD
Defines the library containing the print utility. This DD statement is usually DSNAME=IMS.SDFSRESL,DISP=SHR.

SYSPRINT DD
Defines the system output device to which output is directed. The record format is VBM. If either no block size or a block size less than 141 is specified, the default block size of 141 is assumed. Any block size valid for QSAM and greater than 141 can be specified. Any logical record length can be specified. If no logical record length is specified, the default is 4 less than the block size specified (137 if block size default of 141 is used).

DFSUPRT0 supports output data sets in the cylinder managed area on extended address volumes (EAVs). These data sets exceed 65535 tracks in size, have 3-byte relative track (TTT) extent sizes, and were allocated with DSNTYPE = LARGE on the DD statement. Data sets on EAV volumes can be large data sets that exceed 65535 tracks or smaller ones.

SPOOLnn DD
Describes the spool data set to be printed (where nn is any valid alphanumeric identifier). This DD statement is normally DSNAME=IMS.SYSnn, where 'nn' is assigned by system definition. DCB information either should not be coded or, if coded, must specify RECFM=VBM.

DFSUPRT0 supports input data sets in the cylinder managed area on extended address volumes (EAVs). These data sets exceed 65535 tracks in size, have 3-byte relative track (TTT) extent sizes, and were allocated with DSNTYPE = LARGE on the DD statement.

Return codes

Code
Meaning

0
Successful completion

4
No data sets allocated for printing

8
SYSPRINT DD statement missing

12
I/O error on SYSPRINT data set

Examples of the DFSUPRT0 utility

You can invoke the DFSWT*nnn* procedure as a IMSWTnnn job which prints data sets created by the Spool SYSOUT options.

IMSWT*nnn* member job class and message class are determined by the **CLASS=** and **MSGCLASS=** parameters in the IMSWT000 member in the IMS.SDFSPROC data set.

This job executes procedure DFSWT*nnn*, which invokes the Spool SYSOUT utility program (DFSUPRT0) for printing the Spool SYSOUT data set.

This job must be copied to the IMS.JOBS data set to run.

```
//SPRT0 JOB 1,IMS,CLASS=A,MSGCLASS=A,MSGLEVEL=1
//        EXEC DFSWTnnn
```

Chapter 32. Time-Controlled Operations Verification utility (DFSTVER0)

Use the Time-Controlled Operations (TCO) Verification utility (DFSTVER0) to ensure that your TCO script members are free from errors. If you run the utility before you execute any script member online, the utility detects any script member that would be detected during online execution.

The utility generates reports for the following:

- Errors
- Statistics
- Timer elements (time-schedule requests)
- Messages
- Summaries

Exception: Errors caused by insufficient storage are not detected.

Subsections:

- "Restrictions" on page 587
- "Prerequisites" on page 587
- "Requirements" on page 587
- "Recommendations" on page 587
- "Input and output" on page 587
- "JCL specifications" on page 588
- "Return codes" on page 588

Restrictions

The following restrictions apply to this utility:

You must add TCO script members to the TCO script library before executing the DFSTVER0 utility.

Prerequisites

Currently, no prerequisites are documented for the DFSTVER0 utility.

Requirements

Currently, no requirements are documented for the DFSTVER0 utility.

Recommendations

Currently, no recommendations are documented for the DFSTVER0 utility.

Input and output

You can verify more than one member at a time by assigning an input control statement for each member you are verifying.

© Copyright IBM Corp. 1974, 2024

JCL specifications

The TCO Verification utility is executed as a standard z/OS job. The following are required:

- A JOB statement defined by you
- An EXEC statement
- DD statements

EXEC statement

The region size for execution of the utility is acquired in 4 KB increments for time-schedule request sets and message sets. Each run of the utility, therefore, causes 8 KB to be acquired. Each timing element uses 32 bytes of the 4 KB of storage. The amount of storage messages use varies, depending on the number of segments you specify. In the example job, more time-schedule requests and messages could be added without increasing the storage.

The EXEC statement must be in the form:

```
//  EXEC   PGM=DFSTVER0
```

DD statements

STEPLIB DD
Points to IMS.SDFSRESL, where the TCO modules reside.

SYSPRINT DD
Describes the output data set that contains the reports the utility generates.

SYSUDUMP
Defines the dump data set.

SYSIN DD
Describes the input control data set, which contains the 80-character control statements. The TCO script member names you are verifying with the utility are in columns 1 through 8 of each statement.

The CONT keyword with its parameter can be used to change the size of a message segment; the default segment continuation count is 9. The CONT parameter is a 1- or 2-digit number between 1 and 99 that indicates the new segment continuation count for that particular script.

DFSTCF DD
Points to IMS.TCFSLIB, where the TCO script members reside. You can name the data set TCFSLIB or any other valid dsname.

Return codes

The TCO Verification utility returns a code that indicates the verification processing status. These return codes are:

Code
Meaning

0
No error found in the scripts being verified. The five output reports are generated for each script.

4
Error in the CONT parameter that was specified on the verification JCL.

6
Syntax errors in the script.

8
One of the following errors occurred:

- Unable to get storage
- Unable to open SYSIN data set
- Unable to open SYSPRINT data set
- No DFSTCF DD statement in Verification JCL

10
Error in I/O.

12
Script member not found.

14
Unable to open the script members data set.

If a return code greater than zero is received from the utility, one or more of the scripts being verified has errors.

The DFSTVER0 utility reports are issued for reports that have no errors. Only the error report is issued for scripts that have errors. However, depending on the type of error, it is possible that no error report is generated.

Examples of the DFSTVER0 utility

These examples show how to use the DFSTVER0 utility to create a sample script member, verify members, and create reports.

Sample script member example

The following figure is a sample script member (DFSTCF10).

```
/BRO LTERM CTRL
DFSTCF10 LOADED.                                                    00001500
*TIME      DFSTXITB            S                        ****        00001600
/ASS LTERM LOG27403  TO LINE 31 PTERM 1  ;                          00001700
/START LINE  2 PTERM ALL;                                           00001800
/START LINE 26 PTERM ALL;                                           00001900
/START LINE 18 PTERM ALL;                                           00002000
/STA DB MSDBLM01,MSDBLM02,MSDBLM03,MSDBLM04,MSDBLM05;               00002100
/STA DB MSDBLM06,MSDBLM07,MSDBLM08;                                 00002200
*TIME      DFSTXITB            S                        ****        00002300
/START REGION MSDBMTX3;                                             00002400
/START REGION MSDBMTY3;                                             00002500
/START REGION MSDBMTZ1;                                             00002600
*TIME      DFSTXITB            S                        ****        00002700
PTERM01 BEGIN PTERM1;                                               00002800
PTERM03 BEGIN PTERM3;                                               00002900
/STOP REGION 1;                                                     00003000
*TIME      DFSTXITB            S                        ****        00003100
DFSTCF  LOAD DFSTCF1A;                                              00003200
*TIME      DFSTXITB      0004  S                        ****        00003300
*TIME      DFSTXTIB      0004  S                        ****        00003400
/*
```

Verification example

The following figure is an example of the TCO verification utility procedure of members DFSTCF01, DFSTCF02, and DFSTCF10.

```
//*
//*       THIS JCL IS USED TO VERIFY THE USER SUPPLIED SCRIPTS
//*
//  EXEC  PGM=DFSTVER0
//STEPLIB  DD DSN=IMS.SDFSRESL,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//DFSTCF   DD DSN=IMS.TCFSLIB,DISP=SHR
```

```
//SYSIN    DD *
DFSTCF01 CONT 5
DFSTCF02 CONT 20
DFSTCF10      (ONE OR MORE CARDS SPECIFYING MEMBER NAMES TO BE VERIFIED)
/*
```

In this case, the three script members DFSTCF01, DFSTCF02, and DFSTCF10 are in IMS.TCFSLIB, referred to in the DFSTCF DD statement. The DFSTVER0 utility, DFSTVER0, is found in IMS.SDFSRESL, referred to in the STEPLIB DD statement. The reports generated by the utility are sent to the device you assign for class A output.

Error report example

The following figure is an example of an error report generated by the TCO Verification utility. In the report, the statement sequenced 00003400 (in the DFSTCF10 script member) had a misspelled exit routine name. This exit routine could not be found in the IMS.SDFSRESL library, so it is reported as an error here. The statement is eliminated from the time-schedule request table, which is in the timer elements report.

```
ERROR REPORT FOR MEMBER DFSTCF10

DFS3360E USER EXIT DFSTXTIB REQUESTED NOT FOUND, SEQUENCE NUMBER= 00003400
```

Statistics report example

The following figure shows an example of a statistics report generated by the TCO Verification utility. The only exit routine specified by any time request statement (that was found in IMS.SDFSRESL) is DFSTXITB.

```
 STATISTICS REPORT FOR MEMBER DFSTCF10  PROGRAM EXITS REQUIRED IN IMS.SDFSRESL
DFSTXITB
```

Timer-elements report example

The following figure shows an example of a timer-elements report generated by the TCO Verification utility.

```
 TIMER ELEMENTS REPORT

 TIME OF ACTIVATION EXIT CALLED  ATTRIBUTES   PARM

  STARTUP             DFSTXITB   RES        **** MSG SET
  STARTUP             DFSTXITB   RES        **** MSG SET
  STARTUP             DFSTXITB   RES        **** MSG SET
  STARTUP             DFSTXITB   RES        **** MSG SET
  0957                DFSTXITB   RES SNGL   **** MSG SET
```

The columns in the report are as follows:

Time of Activation
Indicates either STARTUP or the time request is to be processed if this is an actual run.

Exit Called
Indicates the exit to be called for this time request.

Attributes
The following attributes are possible:

RES
Indicates a resident exit routine.

DYN
Indicates a dynamically loaded exit routine.

CONT
Indicates execution each day at the same time.

SNGL
Indicates a single execution.

PARM
Indicates either ****MSG SET, which indicates a message set for this time request, or the 16 bytes of data specified in columns 56 through 71 for this time request in the script member.

Message-table report example

The following figure is an example of a message-table report generated by the TCO Verification utility. The report indicates that the DFSTCF10 script member has five message sets. The asterisk in column 1 indicates a new message. Each message set is separated by a blank line.

```
MESSAGE TABLE REPORT

EACH LINE IS A SEGMENT
* IN COLUMN 1 SIGNIFIES START OF NEW MESSAGE
* IN COLUMN 121 SIGNIFIES SEGMENT IS TRUNCATED

*/BRO LTERM CTRL
 DFSTCF10 LOADED.

*/ASS LTERM LOG27403  TO LINE 31 PTERM 1  ;
*/START LINE  2 PTERM ALL;
*/START LINE 26 PTERM ALL;
*/START LINE 18 PTERM ALL;
*/STA DB MSDBLM01,MSDBLM02,MSDBLM03,MSDBLM04,MSDBLM05;
*/STA DB MSDBLM06,MSDBLM07,MSDBLM08;

*/START REGION MSDBMTX3;
*/START REGION MSDBMTY3;
*/START REGION MSDBMTZ1;

*PTERM01 BEGIN PTERM1;
*PTERM03 BEGIN PTERM3;
*/STOP REGION 1;

*DFSTCF  LOAD DFSTCF1A;
```

A message set is composed of one or more messages. A message set is either single-segment or multi-segment. In the sample report:

- The first message set is a single multi-segment message.
- The second, third, and fourth message sets are multiple single-segment messages.
- The last message set is a single single-segment message.

Summary report example

The following figure is an example of a summary report generated by the TCO Verification utility. The summary report lists the number of time-schedule requests and the number of messages found in a script member. It also summarizes the number of exit routines specified in the time-schedule requests in the member being verified and the amount of storage used.

```
SUMMARY REPORT

# ELEMENTS  # MSGS  #EXIT ROUTINES  STORAGE SIZE
  00005      00014      00001          08324
```

Part 6. Dynamic resource definition utilities

Use the dynamic resource definition (DRD) utilities to perform tasks such as creating a resource definition data set (RDDS), copying the contents from one RDDS into another RDDS or into an IMSRSC repository, and reformatting data to create an RDDS.

© Copyright IBM Corp. 1974, 2024

Chapter 33. Repository to RDDS utility (CSLURP20)

Use the Repository to RDDS utility (CSLURP20) to generate a resource definition data set (RDDS) from an IMSRSC repository. This utility can also be used for backup or migration, or to copy resource and descriptor definitions from one repository to another repository.

The RDDS that is generated is a non-system RDDS that can be used to import the resource and descriptor definitions to IMS with the **IMPORT** command. The resource and descriptor definitions in the generated RDDS can also be used as input to RDDS utilities such as DFSURDD0.

Subsections:

- "Restrictions" on page 595
- "Prerequisites" on page 595
- "Requirements" on page 595
- "Recommendations" on page 595
- "Interfaces" on page 595
- "Input and output" on page 595
- "JCL specifications" on page 596
- "Utility control statements" on page 597
- "Return codes" on page 597

Restrictions

Currently, no restrictions are documented for the CSLURP20 utility.

Prerequisites

Currently, no prerequisites are documented for the CSLURP20 utility.

Requirements

This utility makes connections with Resource Manager (RM) and Structured Call Interface (SCI). An active SCI region must be available on the z/OS image where the utility executes. An active RM region and SCI region must be available on a z/OS image within the sysplex, but it does not have to be the same z/OS image as where the utility executes.

The version of RM must be for IMS Version 12 or later, and it must be enabled for the repository.

Recommendations

Currently, no recommendations are documented for the CSLURP20 utility.

Interfaces

CSLURP20 is an offline batch utility that is started through standard JCL.

Input and output

© Copyright IBM Corp. 1974, 2024

The input to the CSLURP20 utility is a repository that contains the IMS resource and descriptor definitions and the RDDS data set name.

The output from the CSLURP20 utility is a non-system RDDS that contains the resource and descriptor definitions from the repository for the requested IMS system.

The resource and descriptor definitions are overwritten in the specified RDDS data set.

The specified RDDS is formatted before data is written to it, so any existing contents in the RDDS are eventually overwritten. The RDDS is not formatted until all the stored resource definitions are read by the utility from the repository. The RDDS header contains the system type information from the repository.

The utility output contains a summary section that lists the number of resource definitions that are written from the repository.

Messages issued by the utility are written to the SYSPRINT data set. If there is an error in processing the output definitions, an error reading from the repository, or an error writing to the RDDS, the utility terminates with a return code 8.

Errors in the CSLURP20 utility are reported in the SYSPRINT data set with CSL26*xx*E messages.

JCL specifications

You must define a JOB statement, an EXEC statement, and DD statements that define the input and output.

EXEC statement

The format of the EXEC statement is:

```
//S1       EXEC PGM=CSLURP20
```

DD statements

JOBLIB / STEPLIB DD
Points to IMS.SDFSRESL, which contains the executable modules for the utility. Concatenated data sets are not valid.

RDDSDSN DD
Defines the DSN for the RDDS to be used as output from the utility. This DD statement is required. Only one DSN can be specified for the RDDS. Concatenated data sets are not valid.

SYSPRINT DD
Defines the data set to receive messages generated by the utility. This DD statement is required. The DCB parameters for the SYSPRINT data set are RECFM=FB, LRECL=133.

SYSIN DD
Defines the source of the input parameters for the utility. This DD statement is required. The DCB parameters for the SYSIN data set are RECFM=FB, LRECL=80.

For example:

```
//SYSIN          DD     *
IMSPLEX(NAME=PLEX1 IMSID(IMS1))
```

The format of the input parameters for the SYSIN DD statement is as follows:

NAME=
Specifies the IMSplex that the CSLURP20 utility uses for RM and SCI registration. This value is 1 - 5 characters that is appended to the characters CSL, forming the IMSplex name. The

value specified must be the same value specified for IMSPLEX= in the RM and SCI regions. This parameter is required and non-repeatable.

IMSID()
Specifies the IMS ID with the resource and descriptor definitions that the CSLURP20 utility reads from the repository. This parameter is 1 - 4 characters. This parameter is required and non-repeatable.

Utility control statements

Currently, no utility control statements are documented for the CSLURP20 utility.

Return codes

Errors in the CSLURP20 utility are reported in the SYSPRINT data set with CSL26*xx*E messages. Upon completion, the utility terminates with one of the following return codes:

Code	Meaning
0	The utility is completed successfully.
8	The utility encountered a terminating error.

Related concepts
IMSRSC repository administration (System Administration)
Overview of the IMSRSC repository (System Definition)
Maintaining your dynamic resource definition environment (System Definition)
Related tasks
Falling back from using the IMSRSC repository (System Definition)
Importing resource and descriptor definitions from an RDDS by using the IMPORT command (System Definition)
Recovering IMSRSC repository data sets when one or more IMS systems are down but an RM system is active (Operations and Automation)
Creating resource and descriptor definitions in the IMSRSC repository (System Definition)
Related reference
CSL messages (Messages and Codes)

Examples of the CSLURP20 utility

These examples show sample JCL for running the Repository to RDDS utility (CSLURP20), and a summary report example.

JCL example

The following sample JCL can be used to run the CSLURP20 utility.

```
//RPO2RDDS JOB   ,USER,CLASS=A,MSGCLASS=X,NOTIFY=USER
//JOBLIB   DD    DSN=IMSV12.RESLIB
//STEP1    EXEC  PGM=CSLURP20
//SYSUDUMP DD    SYSOUT=*
//SYSPRINT DD    SYSOUT=*
//*
//*************************************************************/
//* SPECIFY A VALID RDDSDSN                                   */
//*************************************************************/
//RDDSDSN  DD   DSN=TEST.NONSYS.RDDS,DISP=OLD
//*
//*************************************************************/
```

```
//* IMSID MUST BE SPECIFIED ON SYSIN
//*************************************************************/
//SYSIN    DD   *
 IMSPLEX(NAME=PLEX1 IMSID(IMS1))
/*
//
```

Sample summary output

The following example shows summary output for the CSLURP20 utility:

```
Sample Output for CSLURP20
--------------------------

CSL2603I CSLURP20 IS PROCESSING RDDS TEST.NONSYS.RDDS
CSL2618I CSLURP20 IS PROCESSING PLEX=CSLPLEX1, IMSID LIST FROM SYSIN IMS1
CSL2620I CSLURP20 SUCCESSFUL REGISTRATION WITH RM, RMNAME=RM1RM
CSL2625I CSLURP20 WRITE TO RDDS SUCCESSFUL FOR RDDSDSN=TEST.NONSYS.RDDS
         FROM REPOTYPE=IMSRSC   REPONAME=IMS_REPOS

   **** SUMMARY ****

DB       COUNT     : 828
DBDESC   COUNT     : 0
PGM      COUNT     : 599
PGMDESC  COUNT     : 0
RTC      COUNT     : 89
RTCDESC  COUNT     : 0
TRAN     COUNT     : 664
TRANDESC COUNT     : 0
```

Related concepts

Overview of dynamic resource definition (System Definition)

Chapter 34. RDDS to Repository utility (CSLURP10)

Use the RDDS to Repository utility (CSLURP10) to copy the contents of a resource definition data set (RDDS) to an IMSRSC repository.

This utility can be used to initially populate or update the repository with resource and descriptor definitions from an RDDS.

When the CSLURP10 utility copies the contents of an RDDS to a repository, resource validation is performed in Resource Manager (RM) as follows:

- When a new transaction or routing code definition is added to the repository, resource validation checking is done to ensure a definition for the associated program either exists in the repository or is being added to the repository with the CSLURP10 utility.
- When an existing program, transaction, or routing code definition is updated in the repository, resource validation checking is done to ensure the attributes being updated do not conflict with the attributes of any associated resource definitions. For example, if a transaction is being modified to Fast Path exclusive FP(E), a check is made to ensure that the associated program is defined as FP(E).

Subsections:

- "Restrictions" on page 599
- "Prerequisites" on page 599
- "Requirements" on page 599
- "Recommendations" on page 600
- "Interfaces" on page 600
- "Input and output" on page 600
- "JCL specifications" on page 600
- "Utility control statements" on page 601
- "Return codes" on page 601

Restrictions

Currently, no restrictions are documented for the CSLURP10 utility.

Prerequisites

Currently, no prerequisites are documented for the CSLURP10 utility.

Requirements

The RDDS can be either a system or non-system RDDS, but must contain a valid set of resource definitions from a successful export operation or from one of the RDDS creation utilities.

This utility makes connections with RM and Structured Call Interface (SCI). An active SCI region must be available on the z/OS image where the utility runs. An active RM region and SCI region must be available on a z/OS image within the sysplex, but it does not have to be the same z/OS image as where the utility runs.

The version of RM must be for IMS Version 12 or later, and it must be enabled for the repository.

© Copyright IBM Corp. 1974, 2024

Recommendations

Currently, no recommendations are documented for the CSLURP10 utility.

Interfaces

The CSLURP10 utility is started through standard JCL.

Input and output

The input to the CSLURP10 utility is a system or non-system RDDS that contains IMS resource and descriptor definitions. The RDDS must contain valid data from a successful export operation or have been produced by one of the RDDS creation utilities.

If a non-system RDDS was created by the Create RDDS from Log Records utility (DFSURCL0) with EXPORTNEEDED to hold only the new or modified resources that were not exported to the IMSRSC repository before IMS was shut down or terminated and IMS needs to cold start, you can specify that non-system RDDS on the RDDS to Repository utility (CSLURP10) to write the definitions to the repository before IMS is cold started.

The output from the CSLURP10 utility is a repository updated with resource definitions copied from the input RDDS.

If there are duplicate resource definitions, which might exist in a non-system RDDS, only the last definition found in the input RDDS are written to the repository.

The utility output contains a summary section that lists the number of resource definitions that are written to the repository and the number of duplicates that are found that are not written to the repository.

Messages issued by the utility are written to the SYSPRINT data set. If there is an error in processing the input definitions, or an error writing to the repository, the utility terminates with a return code 8.

Errors in the CSLURP10 utility are reported in the SYSPRINT data set with CSL26*xx*E messages.

JCL specifications

You must define a JOB statement, an EXEC statement, and DD statements that define the input and output.

EXEC statement

The format of the EXEC statement is:

```
//S1      EXEC PGM=CSLURP10,MEMLIMIT=4G
```

DD statements

JOBLIB / STEPLIB DD
Points to IMS.SDFSRESL, which contains the executable modules for the utility. Concatenated data sets are valid.

RDDSDSN DD
Defines the DSN for the RDDS to be used as input to the utility. This DD statement is required. Only one DSN can be specified for the RDDS. Concatenated data sets are not valid.

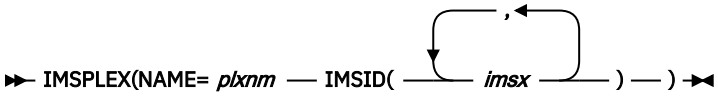
SYSPRINT DD
Defines the data set to receive messages generated by the utility. This DD statement is required. The DCB parameters for the SYSPRINT data set are RECFM=FB, LRECL=133.

SYSIN DD
Defines the source of the input parameters for the utility. This DD statement is required. The DCB parameters for the SYSIN data set are RECFM=FB, LRECL=80.

For example:

```
//SYSIN          DD     *
IMSPLEX(NAME=PLEX1 IMSID(IMS1,IMS2))
```

The format of the input parameters for the SYSIN DD statement is as follows:



NAME=
Specifies the IMSplex that the CSLURP10 utility uses for RM and SCI registration. This value consists of 1 - 5 characters that are appended to the characters CSL, forming the IMSplex name. The value specified must be the same value specified for IMSPLEX= in the RM and SCI regions. This parameter is required and non-repeatable.

IMSID()
Specifies the IMS IDs that the CSLURP10 utility populates in the repository. This parameter is 1 - 4 characters and is repeatable. There can be up to 32 IMS IDs that can be specified. If no IMS ID is specified on the SYSIN DD, CSLURP10 defaults to use the IMS ID in the RDDS header. This parameter is optional.

The resource and descriptor definitions in the RDDS apply to the list of IMS IDs specified in the IMSID keyword. All the IMS systems specified have the same resource and descriptor definitions from the RDDS. If the RDDS is from an MSC system, and multiple IMS IDs are specified, all IMS systems have the same SIDR and SIDL definitions. The IMS of the specified IMS ID does not have to be active when the utility is used to export or write the definitions from the RDDS to the repository.

Utility control statements

Currently, no utility control statements are documented for the CSLURP10 utility.

Return codes

Errors in the CSLURP10 utility are reported in the SYSPRINT data set with CSL26*xx*E messages. On completion, the utility terminates with one of the following return codes:

Code
Meaning

0
The utility is completed successfully.

8
The utility encountered a terminating error. The returned CSL26xxE message provides details on the error.

Related concepts
IMSRSC repository administration (System Administration)
Overview of the IMSRSC repository (System Definition)

Exporting MODBLKS resource and descriptor definitions to an IMSRSC repository (System Definition)
Maintaining your dynamic resource definition environment (System Definition)

Related tasks

Creating resource and descriptor definitions in the IMSRSC repository (System Definition)

Related reference

CSL messages (Messages and Codes)

Examples of the CSLURP10 utility

These examples show sample JCL for running the RDDS to Repository utility (CSLURP10), and a summary report example.

JCL example

The following sample JCL can be used to run the CSLURP10 utility.

```
//RDDS2RPO JOB ,USER,CLASS=A,MSGCLASS=X,NOTIFY=USER
//JOBLIB   DD    DSN=IMSV12.RESLIB
//STEP1    EXEC  PGM=CSLURP10,MEMLIMIT=4G
//SYSUDUMP DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//*
//*****************************************************************/
//* SPECIFY A VALID RDDS DSN FOR INPUT. CONCATENATION IS NOT    */
//* ALLOWED.                                                    */
//*****************************************************************/
//RDDSDSN  DD  DSN=TEST.IMS.RDDS,DISP=SHR
//*
//*****************************************************************/
//* IMSID MAY BE SPECIFIED ON SYSIN OR WILL DEFAULT TO THE      */
//* IMSID ON THE RDDS HEADER RECORD. WHEN USING SYSIN, COMMA    */
//* SEPERATED IMSIDS MAY BE SPECIFIED.                          */
//*****************************************************************/
//SYSIN    DD  *
           IMSPLEX(NAME=PLEX1 IMSID(IMS1))
/*
//
```

Sample summary output

The following example shows summary output for the CSLURP10 utility:

```
Sample Output for CSLURP10
--------------------------

CSL2603I CSLURP10 IS PROCESSING RDDS TEST.IMS.RDDS
CSL2618I CSLURP10 IS PROCESSING PLEX=CSLPLEX1, IMSID LIST FROM SYSIN, IMS1
CSL2620I CSLURP10 SUCCESSFUL REGISTRATION WITH RM, RMNAME=RM1RM
CSL2600I CSLURP10 WRITE TO REPOSITORY WAS SUCCESSFUL FOR REPOTYPE=IMSRSC  , REPONAME=IMS_REPOS

   **** SUMMARY ****

DB       COUNT     : 828
DBDESC   COUNT     : 0
PGM      COUNT     : 599
PGMDESC  COUNT     : 0
RTC      COUNT     : 89
RTCDESC  COUNT     : 0
TRAN     COUNT     : 664
TRANDESC COUNT     : 0
DB       DUPLICATES: 0
DBDESC   DUPLICATES: 0
PGM      DUPLICATES: 0
PGMDESC  DUPLICATES: 0
RTC      DUPLICATES: 0
RTCDESC  DUPLICATES: 0
TRAN     DUPLICATES: 0
TRANDESC DUPLICATES: 0
```

Related concepts

Overview of dynamic resource definition (System Definition)

Chapter 35. Copy RDDS utility (DFSURCP0)

Use the Copy RDDS utility (DFSURCP0) to copy the contents of a resource definition data set (RDDS) into another RDDS.

Control statements are read to supply the IMSID associated with the new RDDS. RDDS copy activities are counted and included in a summary report.

At the beginning of RDDS creation by the DFSURCP0 utility, the RDDS header record is written with a status indicating initialization. If the utility completes successfully, the header is updated to reflect that the RDDS is good, and the time at which the copy was made.

If the IMSID control statement is specified, the RDDS header indicates this IMSID. If this control statement is not specified, the IMSID is retained from the original source RDDS.

If the RETAINTIME control statement is specified, the RDDS header contains the time stamp from the original source RDDS. If this control statement is not specified, the time stamp reflects the time at which the copy was completed.

The JCL to complete these steps can be manually created, or the ISPF panel will automatically generate them if the DFSURCP0 utility is invoked.

Subsections:

- "Restrictions" on page 605
- "Prerequisites" on page 605
- "Requirements" on page 605
- "Recommendations" on page 605
- "Input and output" on page 605
- "JCL specifications" on page 606
- "Utility control statements" on page 606
- "Return codes" on page 607

Restrictions

Currently, no restrictions are documented for the DFSURCP0 utility.

Prerequisites

Currently, no prerequisites are documented for the DFSURCP0 utility.

Requirements

Currently, no requirements are documented for the DFSURCP0 utility.

Recommendations

Currently, no recommendations are documented for the DFSURCP0 utility.

Input and output

© Copyright IBM Corp. 1974, 2024

The input to the DFSURCP0 utility is a preexisting RDDS (the RDDSIN DD statement). Control statements are read from the CONTROL data set.

The new copy of the resource definition records are written to an RDDS data set (the RDDSDSN DD statement). Messages issued by the utility are written to the SYSPRINT data set. A summary report is written to the REPORT data set.

JCL specifications

The DFSURCP0 utility executes as a standard operating system job. You must define a JOB statement, an EXEC statement, and DD statements that define the input and output.

EXEC statement

The format of the EXEC statement is:

```
//S1 EXEC PGM=DFSURCP0
```

DD statements

JOBLIB / STEPLIB DD
Points to IMS.SDFSRESL, which contains the executable modules for the utility. Concatenated data sets are allowed.

RDDSIN DD
Defines the source RDDS that is copied to the target RDDS. This DD statement is required.

RDDSDSN DD
Defines the target RDDS that receives the resource definition records from the source RDDS. This DD statement is required. The DCB parameters for the RDDSDSN data set must be the same as those from the RDDSIN DD statement source RDDS.

SYSPRINT DD
Defines the data set that receives messages generated by the utility. This DD statement is required. The DCB parameters for the SYSPRINT data set are RECFM=FBA, LRECL=133. BLKSIZE must be provided on this DD statement.

REPORT DD
Defines the data set that contains the summary report generated by the utility. This DD statement is required. The DCB parameters for the REPORT data set are RECFM=FBA,LRECL=133. BLKSIZE must be provided on this DD statement.

The summary report includes information such as:

- Images of control statements supplied to the utility
- The data set names associated with the RDDSIN and RDDSDSN DD statements
- Numbers of entries read from the RDDSIN DD statement

CONTROL DD
Defines the source of the input parameters for the utility. This DD statement is required. The DCB parameters for the CONTROL data set are RECFM=FB, LRECL=80.

Utility control statements

IMSID
Specifies the name of the IMS subsystem associated with the RDDS data set generated by the utility. The name is the 4-character ID that is associated with the IMS subsystem.

RETAINTIME
Indicates from where to obtain the time stamp in the RDDS header. If the RETAINTIME control statement is specified, the RDDS header contains the time stamp from the original source RDDS. If this control statement is not specified, the time stamp is the time when the copy was performed.

Return codes

The following return codes are produced:

Code
Meaning

0
The utility completed successfully.

8
The utility encountered a terminating error.

Related concepts
Overview of dynamic resource definition (System Definition)

Examples of the DFSURCP0 utility

These examples show sample JCL for running the DFSURCP0 utility, and a summary report example.

JCL example

The following sample JCL can be used to run the utility.

```
//job name JOB  CLASS=J,MSGCLASS=A,MSGLEVEL=(1,1)
//JOBLIB   DD   DSN=[library data set name],DISP=SHR
//S1       EXEC PGM=DFSURCP0
//RDDSIN   DD   DSN=[RDDS source data set name],DISP=SHR
//RDDSDSN  DD   DSN=[RDDS target data set name],DISP=(,CATLG,DELETE),
//              UNIT=SYSDA,VOL=SER=[Volume name],
//              SPACE=(CYL,(1,1),RLSE),
//              DCB=(LRECL=32756,BLKSIZE=32760,RECFM=VB)
//SYSPRINT DD   SYSOUT=*,
//              DCB=(LRECL=133,BLKSIZE=6118,RECFM=FBA)
//REPORT   DD   SYSOUT=*,
//              DCB=(LRECL=133,BLKSIZE=6118,RECFM=FBA)
//CONTROL  DD   *
IMSID=imid
/*
```

Sample summary output

The following example shows a sample summary report and describes how to interpret the report.

```
* ********************************************************************************
*  RDDS COPY / DFSURCP0                    DATE: 2011/125   TIME: 23:05   PAGE:  1
* ********************************************************************************
*
* CONTROL: IMSID=SYS3
* CONTROL: RETAINTIME
*
*  RDDS INPUT DATA SET NAME :
*   IMSTESTL.IMS1.RDDS1
*
*  RDDS OUTPUT DATA SET NAME:
*   IMSTESTL.IMS1.RDDS2
*
*  IMSID USED . . . . : SYS3
*
*  RDDS COPY TIME . . : 2011.126 05:33:40.605599-UTC
```

```
*  RDDS BLKSIZE . . . : 32760
*
* ******************************************************************************
*       RDDS COPY INFORMATION SUMMARY
*
* TOTAL # OF NON COMMENT CONTROL STATEMENTS READ . . :        2
* TOTAL # OF RDDS RECORDS READ . . . . . . . . . . . :       28
* TOTAL # OF RDDS RECORDS WRITTEN. . . . . . . . . . :       28
* TOTAL # OF RDDS RECORDS UPDATED  . . . . . . . . . :        1
* ******************************************************************************
```

Chapter 36. Create RDDS from Log Records utility (DFSURCL0)

Use the Create RDDS from Log Records utility (DFSURCL0) to create a non-system resource definition data set (RDDS) from checkpoint log records (X'40') and type-2 command records (X'22'). The non-system RDDS can contain all the resource definitions in IMS or a subset of definitions if the STARTTIME, STOPTIME, or EXPORTNEEDED parameter is specified.

Control statements are supplied to the IMSID associated with the RDDS, and the checkpoint ID associated with the checkpoint log records is reformatted to create the RDDS.

A checkpoint begins with a X'4001' record (beginning of checkpoint) and ends with a X'4098' record (end of checkpoint). Only complete checkpoints, where the concatenation of log data sets that contain both of these records, are considered for processing. If the first checkpoint record encountered by the utility is not a X'4001', all checkpoint records are ignored until the first X'4001' record is encountered. If a checkpoint is explicitly selected by the CHKPTID control statement, but the X'4001' record is not present, message DFS3997E is generated and the utility fails. The last checkpoint on a log data set is implicitly selected if no CHKPTID control statement is supplied. If a checkpoint is explicitly or implicitly selected and the X'4098' record for this checkpoint is not found, message DFS3991E is generated and the utility fails.

Time ranges can be specified by STARTTIME and STOPTIME control statements to restrict the part of the log data sets that is considered. If no time ranges are specified, processing begins with the first record in the concatenated log data sets and ends with the last record in the log data sets.

The STARTTIME and STOPTIME parameters are not directly used to specify the checkpoint. They are used as the boundaries of the concatenated log data sets from which checkpoint and type-2 command log records are evaluated. STARTTIME and STOPTIME are specified as Coordinated Universal Time (UTC), and are converted to STCK values. These STCK values do not have to be specified exactly, time stamps are padded to the correct level of precision.

The EXPORTNEEDED parameter can be specified to generate an RDDS with only the resource definitions in the IMS log that have not been exported to the IMSRSC repository.

Definitions for the following resources and descriptors are not written to the RDDS:

- IMS-defined resources and descriptors
- HALDB partitions
- CPIC transactions
- IMS resources created by the DFSINSX0 exit routine that were not created with the export option

The header record and RDDS-formatted records are written to the RDDS data set. Lists of resources, descriptors, and activities performed during the creation of the RDDS are counted and included in a summary report.

At the beginning of RDDS creation by the DFSURCL0 utility, the header record is written with an RDDHD_RDDSTAT value of INIT. If the utility completes successfully, the header is updated to contain an RDDHD_RDDSTAT value of GOOD. If a STOPTIME parameter is specified, the value for RDDS_STCKE reflects that value. Otherwise, the RDDS_STCKE value reflects the Store Time Clock (STCK) value for the last log record that added to the RDDS.

Subsections:

- “Restrictions” on page 610
- “Prerequisites” on page 610
- “Requirements” on page 610
- “Recommendations” on page 610

© Copyright IBM Corp. 1974, 2024

- “Input and output” on page 611
- “JCL specifications” on page 611
- “Utility control statements” on page 612
- “Return codes” on page 613

Restrictions

Currently, no restrictions are documented for the DFSURCL0 utility.

Prerequisites

Currently, no prerequisites are documented for the DFSURCL0 utility.

Requirements

Currently, no requirements are documented for the DFSURCL0 utility.

Recommendations

- Often, a checkpoint does not need to be specified. If one is not specified, the utility selects the last checkpoint in the concatenation of logs.
- If you want to find a specific checkpoint, use the CHKPTID parameter. The CHKPTID value is specified in the format YYDDD/HHMMSS.
- When the STARTTIME and STOPTIME parameters are used, the range of log records to be evaluated is smaller. Use the same checkpoint processing.
- For most processing, the STARTTIME and STOPTIME parameters are not required. The parameters might be useful in the following situations:
  - The STARTTIME parameter might be useful if you have concatenated large log data sets, or multiple log data sets, and want to expedite processing so that the utility can skip data. The parameter is also useful if the explicit checkpoint ID is unknown, but the time before when the checkpoint occurred is known.
  - The STOPTIME parameter might be useful if you want to exclude some X'22' records that occur after the selected checkpoint.

Recommendation: When the EXPORTNEEDED parameter is specified, do not specify STARTTIME or STOPTIME so that all the resource definitions that have not been exported to the IMSRSC repository can be written to the RDDS.

The value specified by the STARTTIME and STOPTIME parameters must be based on STCK values from the suffixes of log records. Do not use time values from the log records other than the suffix STCK because other time values might differ based on their usage, or due to adjustment for leap seconds.

To determine appropriate STCK values, format log records by using the DFSERA30 module or use an equivalent log formatting utility to display STCK values.

The following example shows how output from the DFSERA30 module can be used to determine the STCK value.

```
DFSERA30 - FORMATTED LOG PRINT
22 RECORD - 2008-06-25 20:25:04.339239 UTC <-- STCK value
00000000 000000 00F80000 2208C023 C3D9C540 40404040
00000020 000020 00000000 2008177F 20253133 8995028D
00000040 000040 00000000 00000000 000000A0 00000008
00000060 000060 00000000 00000060 00000000 00000000
```

```
00000080 000080 8CF8C000 000A0700 00FF0000 00020000
000000A0 0000A0 00000000 00000000 00000000 00000000
```

When you specify the EXPORTNEEDED control statement on the Create RDDS from Log Records utility (DFSURCL0), it is recommended that you run the same version of the DFSURCL0 utility as the version of the IMS that produced the IMS logs being used as input. For example, use the IMS 14 DFSURCL0 utility with IMS 14 log data sets. Otherwise, the results can be unpredictable.

Input and output

The input to the DFSURCL0 utility is the IMS log, which contains the log records that are used to create the data in the RDDS. Control statements are read from the CONTROL data set (specified by the CONTROL DD statement).

The DFSURCL0 utility converts the contents of the checkpoint log records (X'4004', X'4006', X'4007', and X'4083') and type-2 command records (X'2208', X'2205', X'2206' and X'2207') into RDDS-formatted records for the corresponding transaction, database, program, and routing code resources. A header and the resource records are written to the RDDS data set (RDDSDSN DD statement). Messages issued by the utility are written to the SYSPRINT data set. A summary report is written to the REPORT data set.

JCL specifications

The DFSURCL0 utility runs as a standard operating system job. You must define a JOB statement, an EXEC statement, and DD statements.

EXEC statement

The format of the EXEC statement is:

```
//S1 EXEC PGM=DFSURCL0
```

DD statements

JOBLIB / STEPLIB DD
Points to the IMS.SDFSRESL data set, which contains the executable modules for the utility. Concatenated data sets are allowed.

SYSUT1 DD
Defines the IMS log data sets that are used as input to the utility. This DD statement is required. Concatenated data sets are allowed.

RDDSDSN DD
Defines the RDDS that receives the resource definition records generated by the utility. This DD statement is required. The DCB parameters for the RDDSDSN data set are LRECL=32756, BLKSIZE=32760, and RECFM=VB.

WORKFILE DD
Defines the data set that holds intermediate report details until they are chosen for printing in the REPORT DD statement. This DD statement is required. The DCB parameters for the WORKFILE data set are RECFM=FBA, LRECL=133.

REPORT DD
Defines the data set that receives messages generated by the utility. This DD statement is required. The DCB parameters for the REPORT data set are RECFM=FBA, LRECL=133.

CONTROL DD
Defines the source of the input parameters for the utility. This DD statement is required. The DCB parameters for the CONTROL data set are RECFM=FB, LRECL=80.

Utility control statements

IMSID
Specifies the name of the IMS subsystem that is associated with the RDDS generated by the utility. The value associated with this parameter is the 4-character ID associated with the IMS subsystem. This parameter is optional. If this parameter is omitted, the IMSID is extracted from the X'4001' checkpoint record.

CHKPTID
Selects the checkpoint that is used as the base for creating the RDDS data set. The checkpoint records associated with this checkpoint ID are processed, and all subsequent X'22' records that follow this checkpoint are processed until the next checkpoint ID, STOPTIME specification, or end of log. This parameter is optional. If it is omitted, the last checkpoint ID that occurs in the log is used.

Values for checkpoint IDs can be found in multiple places, including:

- DFS3804I LATEST RESTART CHKPT messages in the IMS system log are produced each time IMS takes a checkpoint. Extract the checkpoint ID from this message, the syntax for the CHKPTID parameter matches the format of the checkpoint ID in the message
- X'4001' checkpoint records, where the checkpoint ID appears at offset X'0C'.

EXPORTNEEDED
Specifies that a non-system RDDS data set be created from the IMS log and any X'22' log records in the IMS log for any resource definitions that have not yet been exported to the IMSRSC repository.

You can use the output non-system RDDS for one of the following purposes:

- It can be used on the CSLURP10 utility to write the definitions to the IMSRSC repository before an IMS cold start.
- It can be specified on the **IMPORT** command after an IMS cold start from the IMSRSC repository.

If STARTTIME, STOPTIME, or both, are specified with the EXPORTNEEDED control statement, a non-system RDDS is created with only the resource definitions that need to be exported to the repository and also have a create or update time in the specified range of STARTTIME and STOPTIME.

The output non-system RDDS might not contain all the resource definitions in the IMS log and therefore cannot be used for an IMS cold start.

In the case of an error of IMS terminating abnormally after an **EXPORT** command successfully wrote resource definitions to the IMS but IMS did not take a system checkpoint, the resource definitions will remain marked in the IMS log as need to be exported. In this scenario, if IMS must be cold started and the DFSURCL0 utility with EXPORTNEEDED is issued, the RDDS that is generated by the utility will contain the resource definitions from the last successful **EXPORT** command. Here is a timeline of the scenario:

1. You create or update resources and descriptors.
2. You take an IMS checkpoint. Resources that were created in the previous step are written to the IMS log with the "export needed" indicator.
3. You update some more resource definitions.
4. You export resource and descriptor definitions from IMS to the IMSRSC repository by using the command **EXPORT DEFN TARGET(REPO) OPTION(CHANGESONLY)**. All resources that were created or modified in steps “1” on page 612 and “3” on page 612 are written to the repository.
5. IMS ends abnormally before another checkpoint is taken.
6. You run the DFSURCL0 utility with the EXPORTNEEDED control statement. An RDDS is created from the IMS checkpoint in the log at step “2” on page 612 and log records at step “3” on page 612. The resources that have been exported at step “4” on page 612 will still exist in the RDDS created by DFSURCL0.
7. To ensure that there have not been any changes after step “5” on page 612 because IMS had to cold start, you must use the RDDS from step “6” on page 612 to ensure that the resource

definitions in the repository match the resource definitions in the IMS system before it ended abnormally and had to cold start.

NODETAIL
Indicates that the names of the resources modified by data in the X'22' records and are not included in the summary reports. This parameter is optional. If this parameter is omitted, the names of all the modified resources are included in the report.

STARTTIME
Specifies the time stamp that is used to determine where on the log to begin processing. The STCK in the suffix of the log records is used to determine the time on the record. Log records with an earlier time stamp are ignored. Processing begins with the next begin checkpoint record (X'4001') following the STARTTIME specification. If the time stamp occurs in the middle of a checkpoint (after the begin checkpoint record), all subsequent checkpoint records and X'22' records are ignored until the next begin checkpoint record is encountered.

This parameter is optional. If it is omitted, processing begins with the first record in the log. Specify this parameter only if there is a specific reason you want to ignore records before the specified start time.

If both the CHKPTID and STARTTIME parameters are specified, and the STARTTIME specifies a time subsequent to the begin checkpoint record for the specified CHKPTID statement, the indicated checkpoint is not found, and processing terminates with return code 8.

The format of the STARTTIME parameter is a 16-character string: *YYYYDDDHHMMSSTHM*. The STARTTIME value must be specified in UTC. If the data is fewer than 16 characters, it is padded on the right with zeros.

STOPTIME
Specifies the time stamp that is used to determine where on the log to stop processing. The STCK value in the suffix of the log records is used to determine the time on the record. Log records with a later time stamp are ignored. This parameter is optional. If it is omitted, the processing continues until the last record contained in the log.

The format of the STOPTIME parameter is a 16-character string: *YYYYDDDHHMMSSTHM*. The STOPTIME value must be specified in UTC. If the data is fewer than 16 characters, padding on the right occurs:

- Any omitted values in *HHMMSS* are padded, as appropriate, from the value ‘235959'.
- Any other positions are padded with nines.

Return codes

The following return codes are produced:

Code
Meaning

0
The utility completed successfully.

8
The utility encountered a terminating error.

Related concepts
Overview of dynamic resource definition (System Definition)
Related tasks
Recovering IMSRSC repository data sets when one or more IMS systems are down but an RM system is active (Operations and Automation)

Examples of the DFSURCL0 utility

These examples show sample JCL for running the DFSURCL0 utility and a summary report example.

JCL example

The following sample JCL can be used to run the utility:

```
//job name JOB   CLASS=J,MSGCLASS=A,MSGLEVEL=(1,1)
//JOBLIB   DD    DSN=[library data set name],DISP=SHR
//S1       EXEC  PGM=DFSURCL0
//SYSUT1   DD    DSN=[Log data set name(s)],DISP=SHR
//RDDSDSN  DD    DSN=[RDDS data set name],DISP=(,CATLG,DELETE),
//               UNIT=SYSDA,VOL=SER=[Volume name],
//               SPACE=(CYL,(1,1),RLSE),
//               DCB=(LRECL=32756,BLKSIZE=32760,RECFM=VB)
//WORKFILE DD    DSN=[Workfile data set name),DISP=(,CATLG,DELETE),
//               UNIT=SYSDA,VOL=SER=[Volume name],
//               SPACE=(CYL,(1,1),RLSE),
//               DCB=(LRECL=133,BLKSIZE=6118,RECFM=FBA)
//REPORT   DD    SYSOUT=*,
//               DCB=(LRECL=133,BLKSIZE=6118,RECFM=FBA)
//CONTROL  DD    *
IMSID=imid
CHKPTID=yyddd/hhmmss
EXPORTNEEDED
/*
//
```

Summary report example

The following example shows a sample summary report and describes how to interpret the report.

```
* ********************************************************************************
*  RDDS CREATION  / DFSURCL0              DATE: 2011/126   TIME: 11:48   PAGE:  1
* ********************************************************************************
*
* CONTROL: IMSID=SYS3
* CONTROL: CHKPTID=11126/113514
*
*
* RDDS DATA SET NAME:
*  IMSTESTL.IMS1.RDDS2
*
* INPUT LOG DATA SET NAME(S):
*  IMSTESTL.IMS01.OLDSP0
*
* ********************************************************************************
*       RDDS CREATION UTILITY INFORMATION SUMMARY
*
*  LIST OF CHKPTIDS CONTAINED ON LOG:
*  CHKPTID: 11126/113514
*   RDDS DATASET BUILT FROM CHKPTID: 11126/113514
*     CREATED DB       DBX
*     CREATED DB       DBY
*     CREATED DB       DBZ
*     CREATED PGM      PGMX
*     CREATED PGM      PGMY
*     UPDATED PGM      PGMY
*     CREATED TRAN     TRANX
*     CREATED TRAN     TRANY
*     DELETED TRAN     TRANY
*     CREATED RTC      RTCX
*     CREATED RTC      RTCY
*     UPDATED RTC      RTCX
*  CHKPTID: 11126/114444
*
* IMSID USED: SYS3
```

```
*
*
* TIME STAMP ON FIRST LOG RECORD  . . . . . .: 2011126183457995
* TIME STAMP ON LAST LOG RECORD . . . . . .: 2011126184444866
*
* TOTAL # OF NON COMMENT CONTROL RECORDS READ . . . . . .:       2
* TOTAL # OF LOG RECORDS READ . . . . . .:      703
* TOTAL # OF LOG RECORDS SKIPPED. . . . . .:       0
*
* SUPPLIED CHKPTID: 11126/113514 WAS FOUND ON LOG AND PROCESSED
* STATISTICS FOR THIS CHKPTID:
*
* TRANSACTION MODEL : DFSDSTR1
* DATABASE MODEL    : DFSDSDB1
* PGM MODEL         : DFSDSPG1
* ROUTE CODE MODEL  : DBFDSRT1
*
* RESULT OF PROCESSING CHECKPOINT AND TYPE-2 COMMAND RECORDS:
*   TOTAL # OF TRAN RSC  DEFS READ FROM CHKP RECS . . . . .:      668
*   TOTAL # OF TRAN RSC  DEFS CREATED FROM X22 RECS . . . .:        2
*   TOTAL # OF TRAN RSC  DEFS UPDATED FROM X22 RECS . . . .:        0
*   TOTAL # OF TRAN RSC  DEFS DELETED FROM X22 RECS . . . .:        1
*   TOTAL # OF TRAN RSC  DEFS NOT EXPORTED FROM RECORDS . .:        0
*   TOTAL # OF TRAN RSC  DEFS WRITTEN TO RDDS . . . . . .:      669
*
*   TOTAL # OF TRAN DESC DEFS READ FROM CHKP RECS . . . . .:        1
*   TOTAL # OF TRAN DESC DEFS CREATED FROM X22 RECS . . . .:        0
*   TOTAL # OF TRAN DESC DEFS UPDATED FROM X22 RECS . . . .:        0
*   TOTAL # OF TRAN DESC DEFS DELETED FROM X22 RECS . . . .:        0
*   TOTAL # OF TRAN DESC DEFS NOT EXPORTED FROM RECORDS . .:        1
*   TOTAL # OF TRAN DESC DEFS WRITTEN TO RDDS . . . . . .:        0
*
*   TOTAL # OF DB   RSC  DEFS READ FROM CHKP RECS . . . . .:      836
*   TOTAL # OF DB   RSC  DEFS CREATED FROM X22 RECS . . . .:        3
*   TOTAL # OF DB   RSC  DEFS UPDATED FROM X22 RECS . . . .:        0
*   TOTAL # OF DB   RSC  DEFS DELETED FROM X22 RECS . . . .:        0
*   TOTAL # OF DB   RSC  DEFS NOT EXPORTED FROM RECORDS . .:        0
*   TOTAL # OF DB   RSC  DEFS WRITTEN TO RDDS . . . . . .:      839
*
*   TOTAL # OF DB   DESC DEFS READ FROM CHKP RECS . . . . .:        1
*   TOTAL # OF DB   DESC DEFS CREATED FROM X22 RECS . . . .:        0
*   TOTAL # OF DB   DESC DEFS UPDATED FROM X22 RECS . . . .:        0
*   TOTAL # OF DB   DESC DEFS DELETED FROM X22 RECS . . . .:        0
*   TOTAL # OF DB   DESC DEFS NOT EXPORTED FROM RECORDS . .:        1
*   TOTAL # OF DB   DESC DEFS WRITTEN TO RDDS . . . . . .:        0
*
*   TOTAL # OF PGM  RSC  DEFS READ FROM CHKP RECS . . . . .:      605
*   TOTAL # OF PGM  RSC  DEFS CREATED FROM X22 RECS . . . .:        2
*   TOTAL # OF PGM  RSC  DEFS UPDATED FROM X22 RECS . . . .:        1
*   TOTAL # OF PGM  RSC  DEFS DELETED FROM X22 RECS . . . .:        0
*   TOTAL # OF PGM  RSC  DEFS NOT EXPORTED FROM RECORDS . .:        1
*   TOTAL # OF PGM  RSC  DEFS WRITTEN TO RDDS . . . . . .:      606
*
*   TOTAL # OF PGM  DESC DEFS READ FROM CHKP RECS . . . . .:        1
*   TOTAL # OF PGM  DESC DEFS CREATED FROM X22 RECS . . . .:        0
*   TOTAL # OF PGM  DESC DEFS UPDATED FROM X22 RECS . . . .:        0
*   TOTAL # OF PGM  DESC DEFS DELETED FROM X22 RECS . . . .:        0
*   TOTAL # OF PGM  DESC DEFS NOT EXPORTED FROM RECORDS . .:        1
*   TOTAL # OF PGM  DESC DEFS WRITTEN TO RDDS . . . . . .:        0
*
*   TOTAL # OF RTC  RSC  DEFS READ FROM CHKP RECS . . . . .:       94
*   TOTAL # OF RTC  RSC  DEFS CREATED FROM X22 RECS . . . .:        2
*   TOTAL # OF RTC  RSC  DEFS UPDATED FROM X22 RECS . . . .:        1
*   TOTAL # OF RTC  RSC  DEFS DELETED FROM X22 RECS . . . .:        0
*   TOTAL # OF RTC  RSC  DEFS NOT EXPORTED FROM RECORDS . .:        0
*   TOTAL # OF RTC  RSC  DEFS WRITTEN TO RDDS . . . . . .:       96
*
*   TOTAL # OF RTC  DESC DEFS READ FROM CHKP RECS . . . . .:        1
*   TOTAL # OF RTC  DESC DEFS CREATED FROM X22 RECS . . . .:        0
*   TOTAL # OF RTC  DESC DEFS UPDATED FROM X22 RECS . . . .:        0
*   TOTAL # OF RTC  DESC DEFS DELETED FROM X22 RECS . . . .:        0
*   TOTAL # OF RTC  DESC DEFS NOT EXPORTED FROM RECORDS . .:        1
```

```
*   TOTAL # OF RTC  DESC DEFS WRITTEN TO RDDS . . . . . .:        0
*
*   TOTAL # OF RDDSDSN RECORDS WRITTEN  . . . . . .:      217
*   TOTAL # OF RESOURCE DETAIL LINES WRITTEN  . . . . . .:       12
*
* CHECKPOINT LOG RECORDS:
*   TOTAL # OF X4001 LOG RECORDS READ . . . . . .:        1
*
*   SIZE OF X4004 RECORDS (BYTES) . . . . . .:     3944
*   TOTAL # OF X4004 LOG RECORDS READ . . . . . .:       43
*   TOTAL # OF RESOURCE X4004 LOG RECORDS CREATED . . . .:        1
*   TOTAL # OF DESCRIPTOR X4004 LOG RECORDS CREATED . . . .:        0
*   TOTAL # OF X4004 LOG RECORDS DISCARDED  . . . . . .:        1
*   TOTAL # OF TRAN/(SMB) RECORDS WRITTEN . . . . . .:       43
*
*   SIZE OF X4006 RECORDS (BYTES) . . . . . .:      928
*   TOTAL # OF X4006 LOG RECORDS READ . . . . . .:      106
*   TOTAL # OF RESOURCE X4006 LOG RECORDS CREATED . . . .:        1
*   TOTAL # OF DESCRIPTOR X4006 LOG RECORDS CREATED . . . .:        0
*   TOTAL # OF X4006 LOG RECORDS DISCARDED  . . . . . .:        1
*   TOTAL # OF DB/(DDIR) RECORDS WRITTEN  . . . . . .:      106
*
*   SIZE OF X4007 RECORDS (BYTES) . . . . . .:      964
*   TOTAL # OF X4007 LOG RECORDS READ . . . . . .:       56
*   TOTAL # OF RESOURCE X4007 LOG RECORDS CREATED . . . .:        1
*   TOTAL # OF DESCRIPTOR X4007 LOG RECORDS CREATED . . . .:        0
*   TOTAL # OF X4007 LOG RECORDS DISCARDED  . . . . . .:        1
*   TOTAL # OF PGM/(PDIR) RECORDS WRITTEN . . . . . .:       56
*
*   SIZE OF X4083 RECORDS (BYTES) . . . . . .:     1016
*   TOTAL # OF X4083 LOG RECORDS READ . . . . . .:       10
*   TOTAL # OF RESOURCE X4083 LOG RECORDS CREATED . . . .:        1
*   TOTAL # OF DESCRIPTOR X4083 LOG RECORDS CREATED . . . .:        0
*   TOTAL # OF X4083 LOG RECORDS DISCARDED  . . . . . .:        1
*   TOTAL # OF RTC/(RCTE) RECORDS WRITTEN . . . . . .:       10
*
*   TOTAL # OF X4098 LOG RECORDS READ . . . . . .:        1
*
* TYPE-2 COMMAND LOG RECORDS:
*   TOTAL # OF X22 LOG RECORDS READ . . . . . .:       12
*     TOTAL # OF X2202 (DB UPD ACTCTYPE) RECS . . . . . .:        0
*     TOTAL # OF X2205 (DB) RECORDS READ  . . . . . .:        3
*     TOTAL # OF X2206 (PGM) RECORDS READ . . . . . .:        3
*     TOTAL # OF X2207 (RTC) RECORDS READ . . . . . .:        3
*     TOTAL # OF X2208 (TRAN) RECORDS READ  . . . . . .:        3
*     TOTAL # OF X22 IMPORT RECORDS READ  . . . . . .:        0
*       TOTAL # OF IMPORT (DB) RECORDS READ   . . . . . .:        0
*       TOTAL # OF IMPORT (PGM) RECORDS READ  . . . . . .:        0
*       TOTAL # OF IMPORT (RTC) RECORDS READ  . . . . . .:        0
*       TOTAL # OF IMPORT (TRAN) RECORDS READ . . . . . .:        0
*     TOTAL # OF OTHER X22 RECORDS READ . . . . . .:        0
*
* OTHER STATISTICS:
*   TOTAL # OF X45FF LOG RECORDS READ . . . . . .:        2
*
* ******************************************************************************
```

In the summary report example:

- A control statement that contains keyword CHKPTID=11126/113514 is supplied to specify the processed checkpoint.
- The subject log data set is identified.
- The log contained checkpoint 11126/113514 which was used to build the RDDS. The specific resources that were created, updated, or deleted based upon the type-2 command log records are identified.
- The IMSID associated with the log was identified as SYS3.
- The time stamps on the first and last log records read are indicated.
- One control statement was read to direct utility processing.

- 703 log records were read.
- The checkpoint ID specified by the CHKPTID control statement was found on the log and was processed.
- For the indicated checkpoint ID, a breakdown of the significant checkpoint records and their attributes is presented:
  - One X'4001' record was read.
  - For the X'4004', X'4006', X'4007' and X'4083' records, respectively:
    - The size of the records was indicated.
    - The number of records read was indicated.
    - The number of potential RDDS records created to contain new resources, based on data extracted from type-2 command log records, was indicated.
    - The number of potential RDDS records discarded from evaluation was indicated. Records are discarded because:
      - They contain only IMS internally defined resources and descriptors.
      - All of the resources contained on the records were deleted as a result of processing type-2 command log records.
    - The number of RDDS records written for the indicated resource type.
- One X'4098' record was read.
- Statistics related to the type-2 command records were presented:
  - A total of 12 type-2 command log records were read.
  - Based on data contained in the X'2205', X'2206', X'2207' and X'2208' resource records, respectively, the following statistics were indicated:
    - Number of records read
    - Total number of resources written to the RDDS
    - Number of resources created
    - Number of resources updated
    - Number of resources deleted
- Two X'45FF' records (system statistics records) were read.
- 217 records were written to the RDDS (identified by the RDDSDSN DD statement).
- 12 lines identifying the resources created, updated, and deleted were written to the summary report.

Chapter 37. Create RDDS from MODBLKS utility (DFSURCM0)

Use the Create RDDS from MODBLKS utility (DFSURCM0) to extract data from the MODBLKS data set and to reformat the data to create an RDDS.

Control statements are read to identify the IMSID associated with the resource definition data set (RDDS) and the suffix associated with the MODBLKS members. Data from the DFSSMBx, DFSDDIRx, DFSPDIRx, and DFSRCTEx MODBLKS PDS members is formatted for loading into the RDDS. An RDDS header record is created. The header record and checkpoint-like records are written to the RDDS. RDDS creation activities are counted and included in a summary report.

If a transaction has an edit routine assigned to it, for example by the EDIT parameter on the TRANSACT macro, the DFSURCM0 utility identifies the routine by:

- Using a cross-reference file of transaction ID and edit routines created by the DFSURST0 utility
- Extracting information from the IMS nucleus module

Under certain situations, not all IMS subsystems contain all of the MODBLKS member types:

- SMB blocks are evaluated only for DB DC / DCCTL and are required.
- DDIR blocks are evaluated only for DB DC / DBCTL and are required.
- PDIR blocks are evaluated for all system types and are required.
- RCTE blocks are evaluated only for DB DC / DCCTL, but are found only on those systems that support Fast Path.

The nucleus module (DFSVNUCx) is extracted from one of the following three sources, in the following search order:

- SDFSRESL DD statement

  An optional statement that identifies the IMS.SDFSRESL data set that contains the target nucleus if the executable for running the DFSURCM0 utility is in the linklist, such that the JOBLIB and STEPLIB DD statements are not coded in the JCL stream or the subject nucleus is contained in an IMS.SDFSRESL data set other than the one specified in the JOBLIB or STEPLIB DD statements.
- STEPLIB DD statement

  Identifies the IMS.SDFSRESL data set that contains the executable for the DFSURCM0 utility STEPLIB DD statement.
- JOBLIB DD statement

  Identifies the IMS.SDFSRESL data set that contains the executable for the DFSURCM0 utility.

If both the STEPLIB and JOBLIB DD statements are specified, STEPLIB overrides JOBLIB, and the contents of STEPLIB are evaluated.

At the beginning of RDDS creation, the header record is written with an RDDHD_RDDSTAT value of INIT. If the utility completes successfully, the header is updated to contain an RDDHD_RDDSTAT value of GOOD, and the RDDS STCKE value is updated so that it reflects the run time of the utility.

The DFSURCM0 utility can be used to generate an RDDS from system generation macros by adding the following preliminary steps:

- Pre-parse stage 1 to modify parameters and extract subsystem information using the DFSURST0 utility
- Perform a stage 1 IMS MODBLKS generation from IMS system macros to create stage 2
- Allocate work data sets (OBJLIB, MODBLKS)
- Edit stage 2 JCL to reference the work data sets allocated
- Run the generated stage 2 to create a MODBLKS-like data set

© Copyright IBM Corp. 1974, 2024

- Use the MODBLKS-like data set as input to DFSURCM0

The JCL statements that perform these steps can be manually created, or are automatically generated when the "Create RDDS from MODBLKS" ISPF panel is invoked.

You can also use this utility to generate **CREATE** commands from MODBLKS or system generation macros by completing the following tasks:

- Create an RDDS data set from MODBLKS or system generation macros.
- Use the RDDS Extraction utility to generate **CREATE** commands, using the created RDDS and the OUTPUT=CMD control statement as input to the utility.

Subsections:

- “Restrictions” on page 620
- “Prerequisites” on page 620
- “Requirements” on page 620
- “Recommendations” on page 620
- “Input and output” on page 620
- “JCL specifications” on page 620
- “Utility control statements” on page 621
- “Return codes” on page 622

Restrictions

Currently, no restrictions are documented for the DFSURCM0 utility.

Prerequisites

Currently, no prerequisites are documented for the DFSURCM0 utility.

Requirements

Currently, no requirements are documented for the DFSURCM0 utility.

Recommendations

Currently, no recommendations are documented for the DFSURCM0 utility.

Input and output

The input to the DFSURCM0 utility is a MODBLKS or MODBLKS-like data set, with members that contain lists of SMB, DDIR, PDIR, and RCTE resources. Control statements are read from the CONTROL data set.

The DFSURCM0 utility takes the resource definitions in the MODBLKS data set, converts them to the definition format supported by the RDDS, and saves the new definitions in an RDDS. A header and the resource definition records are written to an RDDS (specified by the RDDSDSN DD statement). Messages issued by the utility are written to the SYSPRINT data set. A summary report is written to the REPORT data set.

JCL specifications

The DFSURCM0 utility executes as a standard operating system job. You must define a JOB statement, an EXEC statement, and DD statements that define the input and output.

EXEC statement

The format of the EXEC statement is:

```
//S1 EXEC PGM=DFSURCM0
```

DD statements

JOBLIB / STEPLIB DD
Points to the IMS.SDFSRESL data set, which contains the executable modules for the utility. The data set also contains the nucleus module DFSVNUCx, from which the names for any transaction input edit routines are extracted. Concatenated data sets are allowed.

SDFSRESL DD
Points to the IMS.SDFSRESL data set, which contains the nucleus module DFSVNUCx, from which the names for any transaction input edit routines are extracted. The data set might be the same data set as that is referenced by the JOBLIB or STEPLIB DD statements. The IMS.SDFSRESL data set is an optional data set. If it is omitted, the nucleus module is obtained from the JOBLIB or STEPLIB DD. Only supply this DD statement if the IMS.SDFSRESL data set indicated is different from the one supplied by the JOBLIB or STEPLIB DD statements, or if the executable modules for the DFSURCM0 utility are obtained from the linklist, in which case JOBLIB and STEPLIB DD statements are not supplied.

MODBLKS DD
Defines the MODBLKS data set that is used as input to the utility. This DD statement is required. Concatenated data sets are allowed.

RDDSDSN DD
Defines the RDDS that receives the resource definition records generated by the utility. This DD statement is required. The DCB parameters for the RDDSDSN data set are LRECL=32756, BLKSIZE=32760, RECFM=VB.

SYSPRINT DD
Defines the data set that receives messages generated by the utility. This DD statement is required. The DCB parameters for the SYSPRINT data set are RECFM=FBA, LRECL=133. BLKSIZE must be provided on this DD statement.

REPORT DD
Defines the data set that contains the summary report generated by the utility. This DD statement is required. The DCB parameters for the REPORT data set are RECFM=FBA, LRECL=133. BLKSIZE must be provided on this DD statement.

The summary report includes features and statistics such as:

- Images of control statements supplied to the utility
- Names and sizes of the MODBLKS data set member for each resource type
- Numbers of entries read from the MODBLKS data set for each resource type
- Numbers of records and resources written to the RDDS for each resource type

CONTROL DD
Defines the source of the input parameters for the utility. This DD statement is required. The DCB parameters for the CONTROL data set are RECFM=FB, LRECL=80.

Utility control statements

The following control statements can be specified explicitly, or if the DFSURST0 utility is run before running the DFSURCM0 utility, the control statements generated by DFSURST0 can be used as input to DFURCM0.

If you want to use the control statements generated by DFSURST0, concatenate the data set indicated by the OUTPARMS DD statement in the DFSURST0 JCL as the CONTROL DD statement in the DFSURCM0 utility JCL.

IMSID
Specifies the name of the IMS subsystem that is associated with the RDDS data set generated by the utility. The value associated with this parameter is the 4-character ID associated with the IMS subsystem.

SYSTYPE
Specifies a 1- to 4-character value for the type of IMS subsystem for which the RDDS is used. This parameter is optional. The SYSTYPE default value is DBDC.

Valid SYSTYPE values are:

DBDC
DB/DC system

DB
DBCTL system

DC
DCCTL system

SUFFIX
Specifies a 1-character value for the suffix that is associated with the members in the MODBLKS data set. This parameter can be abbreviated as SUF. This parameter is optional. The SUFFIX default value is 0.

EDITRTN
Specifies a pairing of a transaction ID with the edit routine that is associated with it. Values occur in pairs, with the value for transaction ID and edit routine name separated by a comma, with no intervening spaces. This parameter is optional. For all transaction IDs without an EDITRTN specification, no associated edit routine is identified.

The following example shows the EDITRTN parameter:

```
EDITRTN=ICS,DFSCSMB0
```

ICS is a transaction name, and DFSCSMB0 is an edit routine. Both were specified in stage 1 on the TRANSACT macro as:

```
TRANSACT CODE=ICS,PRTY=(5,12,5),PROCLIM=10,100), MODE=SNGL,EDIT=,DFSCSMB0)
```

Return codes

The following return codes are produced:

Code
Meaning

0
The utility completed successfully.

8
The utility encountered a termination error.

Related concepts
Overview of dynamic resource definition (System Definition)

Examples of the DFSURCM0 utility

These examples show sample JCL for running the DFSURCM0 utility and a summary report example.

JCL example

The following sample JCL statement can be used to run the utility.

```
//job name JOB  CLASS=J,MSGCLASS=A,MSGLEVEL=(1,1)
//JOBLIB   DD   DSN=[library data set name],DISP=SHR
//S1       EXEC PGM=DFSURCM0
//MODBLKS  DD   DSN=[MODBLKS data set name],DISP=SHR
//RDDSDSN  DD   DSN=[RDDS data set name],DISP=(,CATLG,DELETE),
//              UNIT=SYSDA,VOL=SER=[Volume name],
//              SPACE=(CYL,(1,1),RLSE),
//              DCB=(LRECL=32756,BLKSIZE=32760,RECFM=VB)
//SYSPRINT DD   SYSOUT=*,
//              DCB=(LRECL=133,BLKSIZE=6118,RECFM=FBA)
//REPORT   DD   SYSOUT=*,
//              DCB=(LRECL=133,BLKSIZE=6118,RECFM=FBA)
//CONTROL  DD   *
IMSID=imid
SUF=x
/*
```

In the example, because the SDFSRESL DD statement is not supplied, nucleus module DFSVNUCV will be obtained from the data set specified on the JOBLIB DD statement.

Summary report example

The following example shows a sample Summary report and describes how to interpret the report.

```
* ********************************************************************************
*  RDDS BUILD FROM MODBLKS / DFSURCM0     DATE: 2011/125   TIME: 22:33   PAGE: 1
* ********************************************************************************
*
* CONTROL: IMSID=SYS3
* CONTROL: SYSTYPE=DBDC
* CONTROL: SUF=C
*
*  RDDS DATA SET NAME:
*   IMSTESTL.IMS1.RDDS1
*
*  MODBLKS DATA SET NAME:
*   IMSBLD.I12RTS1B.COMBLKS1
*
*  IMSID   USED . . : SYS3
*  SUFFIX  USED . . : C
*  SYSTYPE USED . . : DBDC
*
*  TRAN MEMBER NAME. . . . . . . . : DFSSMB0C
*       SIZE OF MEMBER . . . . . . :   122176
*       SIZE OF MODBLKS BLOCK. . . :      184
*       SIZE OF CHKP BLOCK . . . . :      184
*       SIZE OF RSCX EXTENSION . . :       60
*       TOTAL NUMBER OF BLOCKS . . :      664
*       BLOCKS PER RDDS RECORD . . :      134
*
*  DB   MEMBER NAME. . . . . . . . : DFSDDIRC
*       SIZE OF MEMBER . . . . . . :   139104
*       SIZE OF MODBLKS BLOCK. . . :      168
*       SIZE OF CHKP BLOCK . . . . :       51
*       SIZE OF RSCX EXTENSION . . :       60
*       TOTAL NUMBER OF BLOCKS . . :      828
*       BLOCKS PER RDDS RECORD . . :       64
*
*  PGM  MEMBER NAME. . . . . . . . : DFSPDIRC
```

```
*       SIZE OF MEMBER . . . . . :    67200
*       SIZE OF MODBLKS BLOCK. . . :      112
*       SIZE OF CHKP BLOCK . . . . :       24
*       SIZE OF RSCX EXTENSION . . :       60
*       TOTAL NUMBER OF BLOCKS . . :      599
*       BLOCKS PER RDDS RECORD . . :       88
*
*       * BLOCK FOR DBF#FPU0 PDIR WAS REMOVED
*
*  RTC  MEMBER NAME. . . . . : DFSRCTEC
*       SIZE OF MEMBER . . . . . :     4272
*       SIZE OF MODBLKS BLOCK. . . :       48
*       SIZE OF CHKP BLOCK . . . . :       28
*       SIZE OF RSCX EXTENSION . . :       60
*       TOTAL NUMBER OF BLOCKS . . :       89
*       BLOCKS PER RDDS RECORD . . :       88
*
*  RDDS BUILD TIME : 2011.126 05:33:40.605599-UTC
*  RDDS BLKSIZE    : 32760
*
* ********************************************************************************
*       RDDS BUILD INFORMATION SUMMARY
*
* TOTAL # OF NON COMMENT CONTROL STATEMENTS READ . . :        3
* TOTAL # OF TRANSACT EDIT ROUTINE CONTROL STMTS . . :        0
* TOTAL # OF RDDS RECORDS WRITTEN. . . . . :       29
* TOTAL # OF TRAN ENTRIES IN MODBLKS . . . . . :      664
* TOTAL # OF TRAN RDDS RECORDS WRITTEN . . . . . :        5
* TOTAL # OF DB ENTRIES IN MODBLKS . . . . . :      828
* TOTAL # OF DB RDDS RECORDS WRITTEN . . . . . :       13
* TOTAL # OF PGM ENTRIES IN MODBLKS  . . . . . :      599
* TOTAL # OF PGM RDDS RECORDS WRITTEN  . . . . . :        7
* TOTAL # OF RTC ENTRIES IN MODBLKS  . . . . . :       89
* TOTAL # OF RTC RDDS RECORDS WRITTEN  . . . . . :        2
* ********************************************************************************
```

In this example:

- The IMSID associated with the log was specified on a control statement as SYS3.
- The suffix associated with the modules in the MODBLKS data set was specified on a control statement as C.
- The system type was specified on a control statement as DBDC.
- For each MODBLKS member associated with a resource processed, the following information was presented:
  - The resource type and the name of the associated MODBLKS member that was processed
  - The size of the member was indicated:
    - The size of the records
    - The size of the block for each resource within the MODBLKS member
    - The size of the block for each resource to be contained in an RDDS record
    - The size of the block for the resource extension area associated with each resource in an RDDS record
    - The total number of resource blocks being created for this resource type in the RDDS data set
    - The maximum number of resource blocks in the RDDS records being created for this resource type
- General statistics related to the processing of the utility were presented:
  - Three control statements were read and processed.
  - No control statements identifying transaction edit routines were specified.
  - For each of the four resource types processed from the MODBLKS data set, the following information was presented:

- The number of entries for the indicated resource type that were found in the corresponding MODBLKS member
- The number of RDDS records that were written for this resource type

Chapter 38. DRD IMS SYSGEN stage 1 pre-parser utility (DFSURST0)

Use the DRD IMS SYSGEN stage 1 pre-parser utility (DFSURST0) to reformat or extract selected parameters from the macros contained in the stage 1 source code.

The DFSURST0 utility generates stage 2 JCL after running stage 1 by modifying the parameters on selected stage 1 macro specifications and automates the collection of information about the IMS subsystem.

Control statements are supplied to the utility to direct the modification of the statements in stage 1. Stage 1 source and copy members are read as input to the utility. These elements are combined, processed, and written to an output file, which is subsequently used as input to the system generation process.

The stage 1 source code specifications for the following macros are modified by this utility:

IMSCTRL macro

The SYSTEM statement on the IMSCTRL macro is processed in the following manner:

1. The SYSTEM statement is modified to change the type of IMS generation being performed from the original specification to MODBLKS. The original specification is one of the following types: ALL, BATCH, CTLBLKS, MODBLKS, MSVERIFY, NUCLEUS, ON-LINE
2. The type of IMS system is extracted and formatted as a control statement to be used as input to the DFSURCM0 utility. The type of IMS system is DB/DC, DBCTL, or DCCTL.

There is one occurrence of the IMSCTRL macro in the stage 1 source code.

IMSGEN Macro

The IMSGEN macro is processed in the following manner:

- The MODGEN statement on the IMSGEN macro is modified to specify a data set that contains z/OS compile time system macros. This data set is concatenated to the SYSLIB DD statements in the assemble steps in the stage 2 JCL generated by a successful run of stage 1.
- The OBJDSET statement on the IMSGEN macro is modified to specify a data set that is specified as the SYSLIN DD statements in the assemble steps in the stage 2 JCL generated by a successful run of stage 1.
- The USERLIB statement on the IMSGEN macro is modified to specify a data set that is specified as the USERLIB DD statements in the MODBLKS link edit steps in the stage 2 JCL generated by a successful run of stage 1.
- The second parameter of the NODE statement on the IMSGEN macro is modified to specify the high-level qualifier of the MODBLKS data set that stores the link-edited results of a successful run of stage 2.
- The third parameter of the NODE statement on the IMSGEN macro is modified to specify the high-level qualifier of the following data sets:
  - The ADFSMAC data set, which contains the IMS compile time macros that are concatenated into the SYSLIB DD statements in the assemble steps in the stage 2 JCL generated by a successful run of stage 1
  - The ADFSLOAD data set, which contains IMS bind time routines that are identified by the ADFSLOAD DD statements in the MODBLKS bind step in the stage 2 JCL generated by a successful run of stage 1
- The SUFFIX value for the IMS subsystem is extracted from the IMSGEN macro and is formatted as a control statement that is used as input to the DFSURCM0 utility.

There is one occurrence of the IMSGEN macro in the stage 1 source code.

TRANSACT macro

The TRANSACT macro is processed in the following manner:

© Copyright IBM Corp. 1974, 2024

- The CODE and EDIT statements on the TRANSACT macro are interrogated to determine the list of transactions for which an edit macro is applied.
- The CODE statement determines the list of transaction IDs associated with an occurrence of the TRANSACT macro.
- The EDIT statement determines the edit routine that is associated with the list of transaction IDs indicated on the CODE statement.
- The stage 1 source code can have multiple occurrences of the TRANSACT macro. Although each specification of the TRANSACT macro contains only one CODE specification, multiple transaction ID scans can be associated with this specification. There is only one optional specification of the EDIT statement for a TRANSACT macro specification.

Subsections:

- "Restrictions" on page 628
- "Prerequisites" on page 628
- "Requirements" on page 628
- "Recommendations" on page 628
- "Input and output" on page 628
- "JCL specifications" on page 629
- "Return codes" on page 630

Restrictions

Currently, no restrictions are documented for the DFSURST0 utility.

Prerequisites

Currently, no prerequisites are documented for the DFSURST0 utility.

Requirements

Currently, no requirements are documented for the DFSURST0 utility.

Recommendations

Currently, no recommendations are documented for the DFSURST0 utility.

Input and output

The input to the DFSURST0 utility is the data set that contains the stage 1 macro source for an IMS subsystem. If the stage 1 source references the COPY members, the partitioned data set (PDS) that contains these members is also used as input. Control statements are read from the CONTROL data set.

The DFSURST0 utility reads the stage 1 source, and writes a new version of the stage 1 source that contains modifications to the specification for the selected statements. Control statements are generated that contain data extracted from the stage 1 source that is available for subsequent use by other DRD utilities. A cross reference of transactions and edit routines is created for subsequent use by the DFSURCM0 utility. Messages issued by the utility and a summary report are written to the SYSPRINT data set.

OUTPARMS generated control statements

The following control statements are generated by the DFSURST0 utility for use by the DFSURCM0 utility.

IMSID
The IMSID parameter specifies the name of the IMS subsystem associated with the RDDS data set generated by the utility. The value associated with this parameter is the four-character ID associated with the IMS subsystem.

SYSTYPE
The SYSTYPE parameter specifies the type of IMS subsystem associated with the RDDS data set generated by the utility. It is a one to four-character value.

Valid Values are:

- DB - DBCTL system
- DC - DCCTL system
- DBDC - DB/DC system

SUFFIX
The SUFFIX parameter specifies the suffix associated with the members in the MODBLKS data set. It is a one-character value.

EDITRTN
The EDITRTN parameter specifies a pairing of a transaction ID with the edit routine associated with it. Values occur in pairs, with the value for transaction ID and edit routine name separated by a comma, with no intervening spaces.

JCL specifications

The DFSURST0 utility executes as a standard operating system job. You must define a JOB statement, an EXEC statement, and DD statements that define the input and output.

EXEC statement

The format of the EXEC statement is:

```
//S1 EXEC PGM=DFSURST0
```

DD statements

JOBLIB / STEPLIB DD
Points to the IMS.SDFSRESL data set, which contains the executable modules for the utility. Concatenated data sets are allowed.

STAGE1IN DD
Defines the stage 1 source macro data set that is used as input to the utility. This DD statement is required. If the source is in a PDS member, the member name must be included on the DD statement.

COPYFILE DD
If COPY statements are coded in the stage 1 source identified by the STAGE1IN DD statement, COPYFILE DD defines the PDS that contains stage 1 source macro members. This DD statement is required only if COPY statements are coded.

OUTFILE DD
Defines the data set to receive the modified stage 1 source that is generated by the utility. This DD statement is required. The DCB parameters for the OUTFILE data set are RECFM=FB, LRECL=80. BLKSIZE must be provided on this DD statement.

OUTPARMS DD
Defines the data set that receives the control statements that are generated by the utility. This DD statement is required. The DCB parameters for the OUTPARMS data set are RECFM=FB, LRECL=80. BLKSIZE must be provided on this DD statement.

The data set referenced by this DD statement is used as the CONTROL DD statement in the DFSURCM0 utility JCL stream.

SYSPRINT DD
Defines the data set that receives the messages and the summary report that are generated by the utility. This DD statement is required. The DCB parameters for the SYSPRINT data set are RECFM=FBA, LRECL=133. BLKSIZE must be provided on this DD statement.

CONTROL DD
Defines the source of the input parameters for the utility. This DD statement is required. The DCB parameters for the CONTROL data set are RECFM=FB, LRECL=80.

Utility control statements

MODGEN
Specifies a data set that contains the z/OS compile time system macros that are concatenated into the SYSLIB DD statements in the assemble steps in the stage 2 JCL generated by a successful run of stage 1. The value is the 44-character name associated with this data set. This parameter is optional. If it is omitted, the existing stage 1 specification for MODGEN is used. If there is no existing stage 1 specification, it defaults to SYS1.MODGEN.

USERLIB
Specifies a data set that specifies the USERLIB DD statements in the MODBLKS link-edit steps in the stage 2 JCL generated by a successful run of stage 1. The value is the 44-character name associated with this data set. This parameter is optional. If it is omitted, the existing stage 1 specification for USERLIB is used.

OBJDSET
Specifies a data set that specifies the OBJDSET DD statements in the MODBLKS bind steps in the stage 2 JCL generated by a successful run of stage 1. The value is the 44-character name associated with this data set. This parameter is optional. If it is omitted, the existing stage 1 specification for the second parameter of the NODE statement is used.

MODBLKSHLQ
Specifies the high-level qualifier of the MODBLKS data set that stores the bind results of a successful run of stage 2. The value is the 44-character name associated with this data set. It is mapped to the third parameter of the NODE statement in the stage 1 source generated by the utility.

Attention: This parameter is required to ensure that the system MODBLKS data set is not inadvertently overwritten.

IMSHLQ
Specifies the high-level qualifier of the IMS ADFSMAC and ADFSLOAD data sets that are included in the stage 2 JCL generated by a successful run of stage 1. The value is the part of the 44-character name that precedes the lowest-level qualifier associated with this data set. It is mapped to the third parameter of the NODE statement in the stage 1 source generated by the utility. This parameter is optional. If it is omitted, the existing stage 1 specification for the third parameter of the NODE statement is used.

ASM
Changes the value associated with the ASM statement on the IMSGEN macro in stage 1, which specifies the Assembler JCL to be produced for the stage 2 assembly steps. The value replaces the value associated with the ASM statement specification in stage 1. If multiple parameters are specified, they must be enclosed in parentheses and separated by commas. This parameter is optional. If it is omitted, the existing stage 1 specification for the ASM statement is used.

Return codes

The following return codes are produced:

Code	Meaning
0	The utility completed successfully.
8	The utility encountered a terminating error.

Related concepts

Overview of dynamic resource definition (System Definition)

Examples of the DFSURST0 utility

These examples show sample JCL for running the DFSURST0 utility and a summary report example.

JCL example

The following sample JCL can be used to run the utility.

```
//job name JOB CLASS=J,MSGCLASS=A,MSGLEVEL=(1,1)
//JOBLIB   DD  DSN=[library data set name],DISP=SHR
//S1       EXEC PGM=DFSURST0
//STAGE1IN DD  DSN=[Stage 1 input data set name],DISP=SHR
//COPYFILE DD  DSN=[Copy PDS data set name],DISP=SHR
//OUTFILE  DD  DSN=[Stage 1 output data setname],
//             DISP=(,CATLG,DELETE),
//             UNIT=SYSDA,VOL=SER=[Volume name],
//             SPACE=(CYL,(1,1),RLSE),
//             DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB)
//OUTPARMS DD  DSN=[OUTPARMS control statements data setname],
//             DISP=(,CATLG,DELETE),
//             UNIT=SYSDA,VOL=SER=[Volume name],
//             SPACE=(TRK,(1,1),RLSE),
//             DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB)
//SYSPRINT DD  SYSOUT=*,
//             DCB=(LRECL=133,BLKSIZE=6118,RECFM=FBA)
//CONTROL DD *
     MODGEN=SYS1.MACLIB
     USERLIB=USERID.DFSURST0.USERLIB
     OBJDSET=USERID.DFSURST0.OBJDSET
     MODBLKSHLQ=USERID.DFSURST0
     IMSHLQ=IMSBLD.HMK1010
//*
//
```

In this example, because the ASM parameter is not supplied, the existing stage 1 specification for the ASM statement is used.

Summary report example

The following example shows a sample Summary report and describes how to interpret the report.

```
DFSURST0 CONTROL:    MODGEN=SYS1.MACLIB
DFSURST0 CONTROL:    USERLIB=USERID.DFSURST0.USERLIB
DFSURST0 CONTROL:    OBJDSET=USERID.DFSURST0.OBJDSET
DFSURST0 CONTROL:    MODBLKSHLQ=USERID.DFSURST0
DFSURST0 CONTROL:    IMSHLQ=IMSBLD.HMK1010
COPY MEMBER ARSALL  PROCESSED
ORIGINAL IMSCTRL: ALL
     RECORDS PROCESSED . . .:   13
COPY MEMBER SAMPDBD PROCESSED
     RECORDS PROCESSED . . .:   15
COPY MEMBER IVP#DBD PROCESSED
     RECORDS PROCESSED . . .:    8
COPY MEMBER GENIDENT PROCESSED
     RECORDS PROCESSED . . .:    8
  DFSURST0: STAGE1 PRE-PARSER UTILITY: SUMMARY REPORT
                    DATE: 2008/059 TIME: 16:46
```

```
INPUT LOG DATA SET NAME(S)
USERID.STAGE1(ARSS1 )
NUMBER OF CONTROL RECORDS READ . . . .:       5
NUMBER OF STAGE1 RECORDS READ . . . .:     1132
NUMBER OF COPY STATEMENTS READ . . . .:       4
NUMBER OF LINES FROM COPY MEMEBERS . .:      44
NUMBER OF EXCLUDED JCL STATEMENTS. . .:       0
NUMBER OF STAGE1 COMMENTS IGNORED. . .:     216
NUMBER OF STAGE1 RECORDS WRITTEN . . .:    1177
NUMBER OF IMSCTRL STATEMENTS . . . . .:       1
NUMBER OF TRANSACT STATEMENTS. . . . .:       7
NUMBER OF IMSGEN STATEMENTS. . . . . .:       1
NUMBER OF OUTPARM RECORDS WRITTEN. . .:       3
```

In this example:

- Images of the control statements that were supplied to the utility are displayed.
- The name of each COPY member that was read from the file indicated on the COPYFILE DD statement, and the number of records read from the member, are displayed.
- Although it is always changed to MODBLKS by the utility, the original specification for the TYPE parameter associated with the SYSTEM keyword on the IMSCTRL macro is ALL.
- The name of the data set that contains stage 1 system generation macros is member ARSS1 in data set USERID.STAGE.
- Five control statements were read and processed.
- 1132 records were read from the stage 1 data set member identified.
- 4 COPY members were read from the file indicated on the COPYFILE DD statement.
- 44 records were read from the COPY members.
- The contents of the stage 1 data set member contained only stage 1 macros. No JCL was included in the source. If there were any JCL in the source, such that the stage 1 macros were embedded following a `//SYSIN DD *` statement, the JCL would be excluded from the modified stage 1 source created by the utility.
- 216 comments were found in the stage 1 source. Although these were ignored for processing, they are included in the modified stage 1 source created by the utility.
- 1177 records were written as the modified source.
- 1 IMSCTRL macro statement was found in the input stage 1 source.
- 7 TRANSACT macro statements were found in the input source.
- 1 IMSGEN macro statement was found in the input source.
- 3 records, for potential subsequent use as control statements for the DFSURCM0 utility, were written to the file identified on the OUTPARM DD statement.

Chapter 39. RDDS Extraction utility (DFSURDD0)

Use the RDDS Extraction utility to convert the stored resource and descriptor definitions in a resource definition data set (RDDS) to IMS stage 1 macro statements or IMS type-2 **CREATE** commands. The RDDS Extraction utility can also be used to generate a report that lists the contents of the RDDS.

The RDDS Extraction utility is an offline batch utility.

Subsections:

- "Restrictions" on page 633
- "Prerequisites" on page 633
- "Requirements" on page 633
- "Recommendations" on page 633
- "Input and output" on page 633
- "JCL specifications" on page 634
- "Utility control statements" on page 634
- "Return codes" on page 635

Restrictions

Currently, no restrictions are documented for the DFSURDD0 utility.

Prerequisites

Currently, no prerequisites are documented for the DFSURDD0 utility.

Requirements

Currently, no requirements are documented for the DFSURDD0 utility.

Recommendations

Currently, no recommendations are documented for the DFSURDD0 utility.

Input and output

The input to the RDDS Extraction utility is an RDDS data set that contains resource and descriptor definitions for an IMS. The RDDS must contain valid data from a successful export operation or be produced by one of the Create RDDS utilities.

The generated stage 1 macro statements and the **CREATE** commands are written to the SYSOUT data set. Messages that are issued by the utility are written to the SYSPRINT data set.

When you convert resource definitions to stage 1 macro statements, the output of the utility is DATABASE, APPLCTN, RTCODE and TRANSACT macro statements that represent the resource definitions in the RDDS. Resource descriptor definitions are not converted to stage 1 macro statements.

When you convert resource definitions to IMS type-2 **CREATE** commands, the output of the RDDS Extraction utility is **CREATE DB**, **CREATE DBDESC**, **CREATE PGM**, **CREATE PGMDESC**, **CREATE RTC**, **CREATE RTCDESC**, **CREATE TRAN**, and **CREATE TRANDESC** commands.

© Copyright IBM Corp. 1974, 2024

If you specify the LANG=ASSEM parameter on the APPLCTN macro or the **CREATE PGM** command and the program definition is exported to an RDDS, the utility always generates LANG=COBOL on the APPLCTN macro statement or LANG(COBOL) on the **CREATE PGM** command. The ASSEM and COBOL program languages are treated the same by IMS.

In addition to converting resource and descriptor definitions to stage 1 macros or type-2 **CREATE** commands, the utility can be used to generate a report that lists the contents of the RDDS. In addition to the fields that are included when stage 1 macro statements and **CREATE** commands are generated, specification of the OUTPUT=QUERY control statement in the SYSIN DD statement extracts the following additional fields:

- The last time the resource or descriptor definition was accessed, created, updated, or imported
- The RDDS header information, including the RDDS data set name, status, type, and time-stamp information

This report is written to the SYSOUT data set.

JCL specifications

The RDDS Extraction utility executes as a standard operating system job. You must define a JOB statement, an EXEC statement, and DD statements that define the input and output.

EXEC statement

The format of the EXEC statement is:

```
//S1       EXEC PGM=DFSURDD0,MEMLIMIT=12G
```

The RDDS Extraction utility obtains 64-bit storage. The MEMLIMIT parameter must be specified on the EXEC statement to ensure the total size of usable virtual storage above the bar is adequate. Set the MEMLIMIT parameter to 12 GB or higher.

DD statements

JOBLIB / STEPLIB DD
Points to the IMS.SDFSRESL data set, which contains the executable modules for the utility. This DD statement is required. Concatenated data sets are allowed.

RDDSDSN DD
Defines the RDDS that is used as input to the utility. This DD statement is required. Only one data set can be specified for the RDDS. Concatenated data sets are not allowed.

SYSOUT DD
Defines the data set that receives the output that is generated by the utility. This DD statement is required. The DCB parameters for the SYSOUT data set are LRECL=80, RECFM=FB.

The SYSOUT can be the IMS stage 1 macro statements, IMS type-2 **CREATE** commands, or the Query report.

SYSPRINT DD
Defines the data set that receives the messages generated by the utility. This DD statement is required. The DCB parameters for the SYSPRINT data set are RECFM=FB, LRECL=133.

SYSIN DD
Defines the source of the input parameters for the utility. This DD statement is required. The DCB parameters for the SYSIN data set are RECFM=FB, LRECL=80.

Utility control statements

Output
The OUTPUT parameter specifies the format of the generated output. This parameter is required. The following values are valid for the OUTPUT parameter:

MAC
Converts the data in the RDDS to IMS stage 1 macro statements.

CMD
Converts the data in the RDDS to type-2 CREATE commands.

QUERY
Generates the data in the RDDS as a report.

The Query report is divided into the following sections:

- A display of the contents of the RDDS Header
- A display of the attributes for each resource described in the RDDS
- A count of the total number of entries for each resource type
- A count of the total number of duplicate entries for each resource type

Return codes

The following return codes are produced:

Code
Meaning

0
The utility completed successfully.

4
The utility completed successfully; however, either no transactions were defined for an application or no route codes were defined for a Fast Path-exclusive application.

8
An error occurred trying to open the SYSPRINT data set.

12
An error occurred trying to open the SYSOUT data set.

16
An error occurred trying to read the SYSIN data set.

20
An error occurred trying to parse the SYSIN data.

24
An error occurred trying to obtain storage using the MVS GETMAIN macro.

28
An error occurred trying to obtain a buffer from 64-bit storage.

32
An RDJFCB request failed to return the dsname of the RDDS data set.

36
More than one data set is defined for the RDDS.

40
An error occurred trying to open the RDDS data set that is defined on the RDDSDSN DD statement.

44
An error occurred trying to read the RDDS data set that is defined on the RDDSDSN DD statement.

48
The data set that is defined on the RDDSDSN DD statement is not an RDDS.

52
The RDDS that is defined on the RDDSDSN DD statement does not contain data from a successful export operation.

56
An error occurred trying to start the BPE limited function services (LFS).

Related concepts

Overview of dynamic resource definition (System Definition)

Examples for the DFSURDD0 utility

The following examples show two sets of sample JCL statements for running the RDDS Extraction utility and a sample Query report.

JCL example with output set to MAC

The following sample JCL can be used to generate this utility with IMS stage 1 macro statements specified as the output.

```
//job name JOB  CLASS=J,MSGCLASS=A,MSGLEVEL=(1,1)
//JOBLIB   DD   DSN=library data set name,DISP=SHR
//S1       EXEC PGM=DFSURDD0,MEMLIMIT=12G
//RDDSDSN  DD   DSN=RDDS data set name,DISP=SHR
//SYSOUT   DD   DSN=output data set name,DISP=(,CATLG,DELETE),
//              UNIT=SYSDA,VOL=SER=Volume name,
//              SPACE=(CYL,(1,1),RLSE),
//              DCB=(LRECL=80,RECFM=FB,BLKSIZE=800)
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
OUTPUT=MAC
/*
//
```

JCL example with output set to QUERY

The following sample JCL can be used to generate this utility with the Query report specified as the output.

```
//job name JOB  CLASS=J,MSGCLASS=A,MSGLEVEL=(1,1)
//JOBLIB   DD   DSN=library data set name,DISP=SHR
//S1       EXEC PGM=DFSURDD0,MEMLIMIT=12G
//RDDSDSN  DD   DSN=RDDS data set name,DISP=SHR
//SYSOUT   DD   DSN=output data set name,DISP=(,CATLG,DELETE),
//              UNIT=SYSDA,VOL=SER=Volume name,
//              SPACE=(CYL,(1,1),RLSE),
//              DCB=(LRECL=80,RECFM=FB,BLKSIZE=800)
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
OUTPUT=QUERY
/*
//
```

Sample Query report

The following sample Query report is provided if OUTPUT=QUERY is specified in the SYSIN DD statement.

```
RDDS HEADER RECORD                                                +
     HEADER_LENGTH(168) VERSION(1) STATUS(GOOD)                   +
     IMSID(SYS3) IMSTYPE(DBDC) SYSTEM_RDDS?(Y)                    +
     TIMESTAMP(2008.058 21:50:07.695470-UTC)                      +
     data set_NAME(USERID.TEST.RDDS2                            )
DB NAME(AUTODB) ACCTYPE(UPD) RESIDENT(N) GLOBAL DMB(0000) +
     LOCAL DMB(0001) MODELNAME() MODELTYPE() TMCR(2007.311 16:18:42.49-UTC) +
     TMAC() TMUP() TIMP()
```

```
DB NAME(AUTODBH) ACCTYPE(UPD) RESIDENT(N) GLOBAL DMB(0000) +
     LOCAL DMB(0002) MODELNAME() MODELTYPE() TMCR(2007.311 16:18:42.49-UTC) +
     TMAC() TMUP() TIMP()
DB NAME(BANKATMS) ACCTYPE(EXCL) RESIDENT(N) GLOBAL DMB(0000) +
     LOCAL DMB(0003) MODELNAME() MODELTYPE() TMCR(2007.311 16:18:42.49-UTC) +
TMAC() TMUP() TIMP()
PGM NAME(EMHPSB2) BMPTYPE(N) DOPT(N) FP(E)GPSB(N) +
     RESIDENT(N) SCHDTYPE(PARALLEL) TRANSTAT(N) MODELNAME() +
     MODELTYPE() TMCR(2008.354 22:17:41.80-UTC) TMAC() +
     TMUP() TIMP()
TRAN NAME(EMHTX2) AOCMD(N) CLASS(1) CMTMODE(SNGL) +
     CONV(N) DCLWA(Y) DIRROUTE(N) EDITUC(Y) FP(E) INQ(N) +
     LPRI(1) MAXRGN(0) MSGTYPE(SNGLSEG) NPRI(1) PARLIM(65535) +
     PGM(EMHPSB2) PLCT(65535) PLCTTIME(6553500) RECOVER(Y) +
     REMOTE(N) RESP(Y) SEGNO(0) SEGSZ(0) SERIAL(N) +
     TRANSTAT(N) WFI(N) MODELNAME() MODELTYPE() TMCR(2008.354 22:17:42.74-UTC) +
     TMAC() TMUP() TIMP()
TRAN NAME(EMHTX3) AOCMD(N) CLASS(1) CMTMODE(SNGL) +
     CONV(N) DCLWA(Y) DIRROUTE(N) EDITUC(Y) FP(E) INQ(N) +
     LPRI(1) MAXRGN(0) MSGTYPE(SNGLSEG) NPRI(1) PARLIM(65535) +
     PGM(EMHPSB2) PLCT(65535) PLCTTIME(6553500) RECOVER(Y) +
     REMOTE(N) RESP(Y) SEGNO(0) SEGSZ(0) SERIAL(N) +
     TRANSTAT(N) WFI(N) MODELNAME() MODELTYPE() TMCR(2008.354 22:17:42.74-UTC) +
     TMAC() TMUP() TIMP()
* RTC NAME(EMHTX2) INQ(N) PGM(EMHPSB2) MODELNAME() MODELTYPE() +
     TMCR(2008.354 22:17:42.73-UTC) TMAC() TMUP() +
     TIMP()
RTC NAME(EMHTX22) INQ(N) PGM(EMHPSB2) MODELNAME() +
     MODELTYPE() TMCR(2008.354 22:17:42.73-UTC) TMAC() +
     TMUP() TIMP()
* RTC NAME(EMHTX3) INQ(N) PGM(EMHPSB2) MODELNAME() MODELTYPE() +
     TMCR(2008.354 22:17:42.73-UTC) TMAC() TMUP() +
     TIMP()
RTC NAME(EMHTX32) INQ(N) PGM(EMHPSB2) MODELNAME() +
     MODELTYPE() TMCR(2008.354 22:17:42.73-UTC) TMAC() +
     TMUP() TIMP()
TRAN NAME(TSTAD2R2) AOCMD(N) CLASS(1) CMTMODE(MULT) +
     CONV(N) DCLWA(Y) DIRROUTE(N) EDITUC(Y) INQ(N) +
     LCT(65535) LPRI(1) MAXRGN(0) MSGTYPE(MULTSEG) +
     NPRI(1) PARLIM(65535) PGM(AD2TP) PLCT(65535) +
     PLCTTIME(6553500) RECOVER(Y) REMOTE(N) RESP(Y) +
     SEGNO(0) SEGSZ(0) SERIAL(N) TRANSTAT(N) WFI(N) +
     MODELNAME() MODELTYPE() TMCR(2007.319 18:50:47.98-UTC) +
     TMAC() TMUP() TIMP(2008.012 01:13:32.20-UTC)
PGM NAME(APOL1) BMPTYPE(N) DOPT(Y) FP(N)GPSB(N) RESIDENT(N) +
     SCHDTYPE(SERIAL) TRANSTAT(N) MODELNAME() MODELTYPE() +
     TMCR(2007.319 18:50:47.28-UTC) TMAC() TMUP(2008.002 22:50:56.69-UTC) +
     TIMP(2008.012 01:13:32.20-UTC)
PGM NAME(FPPSB02) BMPTYPE(N) DOPT(N) FP(E)GPSB(N) +
     RESIDENT(Y) SCHDTYPE(PARALLEL) TRANSTAT(N) MODELNAME() +
     MODELTYPE() TMCR(2007.311 16:18:42.48-UTC) TMAC() +
     TMUP() TIMP()
**** SUMMARY ****
TRAN     COUNT :     0
TRANDESC COUNT :     0
DB       COUNT :     2045
DBDESC   COUNT :     0
PGM      COUNT :     5506
PGMDESC  COUNT :     0
RTC      COUNT :     0
RTCDESC  COUNT :     0

TRAN     DUPLICATES: 0
TRANDESC DUPLICATES: 0
DB       DUPLICATES: 0
DBDESC   DUPLICATES: 0
PGM      DUPLICATES: 0
PGMDESC  DUPLICATES: 0
RTC      DUPLICATES: 0
RTCDESC DUPLICATES: 0
```

In the Query report example:

- For each of the resource definitions and resource descriptor types, the following counts are displayed:
  - The number of resources contained in the RDDS.
  - The number of resource entries that have the same resource name as another entry in the RDDS.
- Two types of routing code-type records are included in the report:
  - Routing codes that are the result of explicitly being included in the stage 1 input or **CREATE** command
  - Routing codes that are associated with Fast Path-exclusive transactions. This type of routing code is preceded by an asterisk (*) in the report.

Notices

This information was developed for products and services offered in the US. This material might 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 grant 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 US

For license inquiries regarding double-byte character set (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

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 jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM 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 provide 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 Director of Licensing IBM Corporation North Castle Drive, MD-NC119 Armonk, NY 10504-1785 US

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 document 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.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

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

© Copyright IBM Corp. 1974, 2024

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.

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 actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate 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_.

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.

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 "Copyright and trademark information" 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.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

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.

To learn more, see IBM Privacy Statement.

Bibliography

This bibliography lists all of the publications in the IMS 15.5 library.

Title	Acronym
IMS Version 15.5 Application Programming	APG
IMS Version 15.5 Application Programming APIs	APR
IMS Version 15.5 Commands, Volume 1: IMS Commands A-M	CR1
IMS Version 15.5 Commands, Volume 2: IMS Commands N-V	CR2
IMS Version 15.5 Commands, Volume 3: IMS Component and z/OS Commands	CR3
IMS Version 15.5 Communications and Connections	CCG
IMS Version 15.5 Database Administration	DAG
IMS Version 15.5 Database Utilities	DUR
IMS Version 15.5 Diagnosis	DGR
IMS Version 15.5 Exit Routines	ERR
IMS Version 15.5 Installation	INS
IMS Version 15.5 Licensed Program Specifications	LPS
IMS Version 15.5 Messages and Codes, Volume 1: DFSMessages	MC1
IMS Version 15.5 Messages and Codes, Volume 2: Non-DFS Messages	MC2
IMS Version 15.5 Messages and Codes, Volume 3: IMSAbend Codes	MC3
IMS Version 15.5 Messages and Codes, Volume 4: IMSComponent Codes	MC4
IMS Version 15.5 Operations and Automation	OAG
IMS Version 15.5 Release Planning	RPG
IMS Version 15.5 System Administration	SAG
IMS Version 15.5 System Definition	SDG
IMS Version 15.5 System Programming APIs	SPR
IMS Version 15.5 System Utilities	SUR

© Copyright IBM Corp. 1974, 2024

Index

Special Characters

/TEST MFS command 168

Numerics

3270 Information Display System
    copy function
        remote terminals 232
    selector pen
        specifying 232

A

ACB (application control block) 3
ACB Generation and Catalog Populate utility (DFS3UACB)
    BUILD statement
        parameters 320
        syntax 319
    control statements
        format 319
        parameters 320
        requirements 319
        syntax 319
    DD statements 317
    DELETE statement
        parameters 320
        syntax 319
    input 315
    JCL 316
    output 315
    overview 313
    prerequisites 314
    recommendations 314
    requirements 314
    restrictions 314
    return codes 327
ACB library
    copying, with IMS catalog 331, 339
ACB Maintenance utility (DFSRRC00)
    ACBGEN procedure 5
    control statements
        BUILD 8
        BUILD DBD 8
        DELETE 8
        format 8
        requirements 7
    description 3
    DFSACBCP control statement 7
    examples 11
    IMS.ACBLIB 3
    input 5
    JCL 5
    output 5
    return codes 11
ACBGEN 3
ACBGEN and Catalog Populate utility (DFS3UACB), *See* ACB Generation and Catalog Populate utility (DFS3UACB)
ACBGEN procedure
    DD statements 6
    EXEC statement 6
ACBLIB data set
    ACB Maintenance utility 3
ACCESS= parameter
    DBD statement 39
accessibility
    features ix
    keyboard shortcuts ix
active region messages
    Log Recovery utility (DFSULTR0) 513
alternate PCB statement, PSB generation 266
Analysis utilities and reports
    Fast Path Log Analysis utility (DBFULTA0) 399
    File Select and Formatting Print utility (DFSERA10) 399
    IMS Monitor Report Print utility (DFSUTR20) 399
    Log Transaction Analysis utility (DFSILTA0) 399
    Offline Dump Formatter utility (DFSOFMD0) 399
    Statistical Analysis utility (DFSISTS0) 399
AOI (Automated Operator Interface)
    IOASIZE requirement 289
Application Control Block (ACB)
    ACBLIB library 3
    maintaining
        control statements 7, 319
        input and output 5
        overview 3
    prerequisites 4
    recommendations 5
    requirements 4
    restrictions 3
Application Control Blocks Maintenance utility 3
application programs
    BMP
        database uncommitted updates restriction 263
    metadata
        ACB Generation and Catalog Populate utility (DFS3UACB) 313
        IMS catalog, loading 373
        IMS catalog, maintenance 365
        loading into IMS catalog 373
archive utility 493
AREA statement
    format 61
    keywords 61
    parameter description 61
    syntax 61
ATTR= operand (DFLD statement)
    parameters
        NODET/DET/IDET 232
        NODISP|HI 233
        NOMOD|MOD 233
        NOPROT/PROT 232
        STRIP|NOSTRIP 233

ATTR= operand (DFLD statement) *(continued)*
parameters *(continued)*
YES, nn 234
specifying 232
with copy lock 232
ATTR= operand (MFLD statement)
specifying 186
attribute data
input message fields
ATTR= operand (MFLD statement) 186
output device fields
ATTR= operand (MFLD statement) 186
specifying 232
attribute simulation
specifying 232
Automated Operator Interface (AOI) 289

B

B= parameter
DFSERA10 OPTION statement 430
backup operations, MFS library 259
batch message processing (BMP)
application programs
database uncommitted updates restriction 263
batch mode (MFS Language utility) 254
Batch SPOC utility (CSLUSPOC)
description 527
examples
JCL example 530
Output example 530
Output example with no wait time specified 530
EXEC statement 528
input and output 527
JCL specifications 528
Parameter keywords 528
prerequisites 527
recommendations 527
requirements 527
restrictions 527
return codes 529
BLOCK= parameter
DATASET statement 53
BMP (batch message processing)
application programs
database uncommitted updates restriction 263

C

C= parameter
C= parameter
File Select and Formatting Print utility 430
catalog
ACB Generation and Catalog Populate utility (DFS3UACB) 313
ACBGEN and population in a single step 313
copying 331, 339
DBRC, without 329, 369
defining 329, 369
IMS catalog utilities 311
loading 373
loading during ACBGEN 313
maintenance 365
catalog *(continued)*
populating 373
populating during ACBGEN 313
utilities
DFS3UCD0 329, 369
catalog database
purging records 389
segments, deleting 389
catalog Directory Recovery utility
DFS3RU00 353
Catalog Record Purge utility 389
catalog utilities
record purge utility 389
checkpoints
system
database uncommitted updates restriction 263
COMP= parameter
ACBGEN procedure 6
compilation statements
COPY 245
EJECT 250
END 250
EQU 246
equate processing 246
PRINT 249
RESCAN 247
RESCAN statement 247
SPACE 250
STACK statement 248
summary of statements 176
SYSIN 172
SYSLIB 172
SYSPRINT 172
TITLE 249
UNSTACK 248
COMPR= operand (DIV statement)
specifying 216
concatenated equates 247
COND= operand (LPAGE statement), specifying 179
COND= parameter
File Select and Formatting Print utility (DFSERA10) 430
control blocks
creation by MFS Language utility 252
MFS Language utility 167
CONTROL statement
File Select and Formatting Print utility (DFSERA10) 425
control statement listing
assembler listing 16
diagnostics 16
copy function
remote terminals 232
COPY option
File Select and Formatting Print utility (DFSERA10) 429
Copy RDDS utility (DFSURCP0)
DD statements 606
description 605
examples
JCL example 607
Sample summary output 607
EXEC statement 606
input and output 605
JCL specifications 606
prerequisites 605
recommendations 605

Copy RDDS utility (DFSURCP0) *(continued)*
requirements 605
restrictions 605
return codes 607
Utility control statements 606
COPY statement
Log Archive utility 499
COPY statement (language utility) 245
copying log records into user data sets 493
Create RDDS from Log Records utility (DFSURCL0)
DD statements 611
description 609
examples
JCL example 614
Summary report example 614
EXEC statement 611
input and output 611
JCL specifications 611
prerequisites 610
recommendations 610
requirements 610
restrictions 610
return codes 613
Utility control statements 612
Create RDDS from MODBLKS utility (DFSURCM0)
DD statements 621
description 619
examples
JCL example 623
Summary report example 623
EXEC statement 621
input and output 620
JCL specifications 620
prerequisites 620
recommendations 620
requirements 620
restrictions 620
return codes 622
Utility control statements 621
CSLULALE (OM Audit Trail Format and Print module)
DD statements 452
description 451
JCL specifications 451
Limiting log data to a specified time range 452
Utility control statements 452
CSLURP10 (RDDS to Repository utility)
DD statements 600
description 599
examples
JCL example 602
Sample summary output 602
EXEC statement 600
input and output 600
JCL specifications 600
prerequisites 599
recommendations 600
requirements 599
restrictions 599
return codes 601
utility control statements 601
CSLURP20 (Repository to RDDS utility)
DD statements 596
description 595
examples
CSLURP20 (Repository to RDDS utility) *(continued)*
examples *(continued)*
JCL example 597
Sample summary output 598
EXEC statement 596
input and output 595
JCL specifications 596
prerequisites 595
recommendations 595
requirements 595
restrictions 595
return codes 597
Utility control statements 597
CSLUSPOC (Batch SPOC utility)
description 527
examples
JCL example 530
Output example 530
Output example with no wait time specified 530
EXEC statement 528
input and output 527
JCL specifications 528
Parameter keywords 528
prerequisites 527
recommendations 527
requirements 527
restrictions 527
return codes 529

D

D= keyword
control statements
DFSERA10 CONTROL 426
DFSERA10 OPTION 432
Data Capture exit routine
EXIT= parameter 43
data entry database (DEDB)
DBD generation 28
Database Description (DBDs)
generating
DEDB databases 22
GSAM databases 20
HSAM databases 19
SHSAM databases 19
Database Description Generation utility
control statements, input order 24
database description rules, DBD generation 27
Database Descriptions (DBDs)
generating
HDAM and PHDAM databases 21
HDAM databases 21
HIDAM and PHIDAM databases 21
HISAM databases 20, 21
Index and PSINDEX databases 23
logical segment types 24
MSDBs 22
PHDAM databases 21
SHISAM databases 20
Database Recovery Control utility (DSPURX00)
description 531
examples 534
invoking the utility 534
prerequisites 531

Database Recovery Control utility (DSPURX00) *(continued)*
recommendations 531
requirements 531
restrictions 531
databases
IMS catalog
copying 331, 339
metadata
ACB Generation and Catalog Populate utility (DFS3UACB) 313
IMS catalog, loading 373
IMS catalog, maintenance 365
loading into IMS catalog 373
DATASET statement
database
GSAM 49
HDAM 49
HIDAM 50
HISAM 50
HSAM 51
INDEX 51
LOGICAL 52
MSDB 52
description 48
format 49
keywords 52
parameter description 52
DATXEXIT parameter
DBD statement 47
DBD (Database Description) generation
AREA statement
description 61
format 61
keywords 61
syntax 61
assembler listings 16
block size, specifying minimum for databases 53
coding conventions 27
control interval size, specifying minimum for databases 53
control statement formats
END 140
DATASET statement
description 48
dividing database into multiple data set groups 48
format 49
LABEL field 48
DBD statement 28
DBDGEN statement 140
DD statements 60
DEDB database 28
description rules 27
DFSCASE statement
description 136
keywords 137
parameters 137
DFSMAP statement
description 133
keywords 134
parameters 134
DFSMARSH statement
description 127
keywords 128
parameters 128
DBD (Database Description) generation *(continued)*
diagnostics 16
END statement 140
error conditions 18
examples
Fast Path DEDB 147
Fast Path MSDB 146
Fast Path secondary indexes 155
GSAM 145
HDAM 142
HIDAM 143
HISAM 142
HSAM 141
index generation 140, 144
logical relationships 148
secondary indexes 152
secondary indexing or logical relationships 140
shared secondary indexes 154
Fast Path database 28
Fast Path DEDB 22
Fast Path MSDB 22
FIELD statement
description 101
format 113
keywords 113
FINISH statement 140
GSAM (Generalized Sequential Access Method) 28
GSAM database 20
HDAM database 21, 28
HIDAM database 21, 28
HISAM database 20, 28
HSAM database 19, 28
index generation
logical 24
primary HIDAM 23
secondary index 23
input record structure 26
LABEL field 48
LCHILD statement
defining logical relationships 93
defining primary index relationship 94
defining secondary index relationships 94
description 93
format 97
MSDB database 28
output
assembler listing 16
diagnostics 16
example 17
load module 18
segment flag codes 16
types 16
overview of DBDGEN
consists of 15
control statements 15
databases used with 19
parameter descriptions 28
prerequisites 15
procedure 158
recommendations 15, 157
requirements 15
restrictions 15
SEGM statement
description 63

DBD (Database Description) generation *(continued)*
SEGM statement *(continued)*
keyword abbreviations 78
keywords 78
pointer keyword options and abbreviations 81
SHISAM database 20
SHSAM database 19
specifying options for DEDBs
DBDGEN statement 140
END statement 140
FINISH statement 140
summary of statement types 24
XDFLD statement
description 123
format 123
keywords 123
DBD generation input record structure (non-DEDB)
exception 25
requirement 26
DBD library
copying, with IMS catalog 331, 339
DBD= keyword
ACB Generation and Catalog Populate utility (DFS3UACB) 320
ACB Maintenance utility 8
DBDGEN
procedure
JCL parameters 158
DBDGEN utility
control statements, input order 24
DBFULTA0 (Fast Path Log Analysis utility)
error processing 408
Fast Path report types 409
input and output 402
JCL requirements 403
overview 401
prerequisites 402
recommendations 402
reports
Detail-Listing-of-Exception-Transactions 409
Overall Summary of Resource Usage and Contentions 416
Overall Summary of Transit Times 416
Recapitulation-of-the-Analysis 419
Summary-of-Exception-Detail-by-Transaction-Code 415
Summary-of-Region-Occupancy 418
Summary-of-VSO-Activity 419
requirements 402
restrictions 402
utility control statements 404
DBRC (Database Recovery Control) 531
DD statements
DFSUPRT0 585
DD1= parameter
AREA statement 61
DDNAME= keyword
control statements
DFSERA10 CONTROL 426
DFSERA10 OPTION 432
DDNOUT= keyword
DFSERA10 control statement 427
Deadlock reporting
for U777 and U123 abends 439
Deadlock reporting *(continued)*
resultant state of the lock 442
DEDB (data entry database)
defining
DBD generation 22
fields 102
segments 63
naming 28
DEDB (Data Entry Database)
defining
DFSCASE statement 136
DFSMAP statement 133
DFSMARSH statement 127
DFSCASE statement 136
DFSMAP statement 133
DFSMARSH statement 127
DELETE function (MFS Service utility DFSUTSA0) 553
Detail-Listing-of-Exception-Transactions Report
Fast Path Log Analysis utility 409
DEV statement
FEAT= operand 204
FTAB= operand 201
PFK= operand 205
SLDI= operand 208
SLDP= operand 208
specifying 189
SUB= operand 208
VERSID= operand 208
VTAB= operand 207
device characteristics table 198
DFLD (device field statement)
iterative processing 183, 226
PASSWORD parameter 230
printing generated DFLD statements 225
DFS3CCE0
input 332
JCL 334, 343
output 332
prerequisites 332
recommendations 332
requirements 332
restrictions 332
return codes 339
statistics report, export 337
DFS3CCE0 utility
overview 331
DFS3CCI0
input 340
JCL 343
output 340
prerequisites 340
recommendations 340
requirements 340
restrictions 340
return codes 352
statistics report, import 350
DFS3CCI0 utility
overview 331
DFS3CM00
DD statements 366
input 365
output 365
overview 365

DFS3CM00 *(continued)*
requirements 365
restrictions 365
return codes 367
DFS3LU00 357
DFS3PU00
DD statements 336, 378
input 375
JCL 376
output 375
overview 373
prerequisites 374
recommendations 375
requirements 374
restrictions 374
return codes 387
statistics report 385
DFS3PU10 (IMS Catalog Record Purge utility) 389
DFS3RU00 (catalog Directory Recovery utility) 353
DFS3UACB utility
DD statements 317
input 315
JCL 316
output 315
overview 313
prerequisites 314
recommendations 314
requirements 314
restrictions 314
return codes 327
DFS3UCD0 329, 369
DFSACBCP control statement 7
DFSCASE statement
keywords 137
parameters 137
DFSERA10 (File Select and Formatting Print utility)
control statements
COMMENT 432
CONTROL 425
description 425
END 432
OPTION 427
COPY option 429
examples 432
input 423
JCL requirements
DD statements 424
description 424
NEGOF option 429
optional keywords
B= 430
C= 430
COND= 430
D= 432
DDNAME= 432
E= 431
EXITR= 431
FLDLEN= 430
FLDTYP= 430
H= 430
L= 430
O= 429
OFFSET= 429
P= 432
DFSERA10 (File Select and Formatting Print utility) *(continued)*
optional keywords *(continued)*
PARM= 429
PRTSYS= 432
STARTAF= 430
STOPAFT= 430
SYM= 429
T= 430
V= 430
VALUE= 430
output 423
overview 423
prerequisites 423
PRINT option 429
recommendations 423
requirements 423
restrictions 423
DFSERA10 utility modules
DL/I Call Image Capture module (DFSERA50) 438
Enhanced Select module (DFSERA70) 438
IMS Trace Table Record Format and Print module (DFSERA60) 438
OM Audit Trail Format and Print module (CSLULALE) 438
Program Isolation Trace Record Format and Print module (DFSERA40) 438
Record Format and Print module (DFSERA30) 438
DFSERA30 (Record Format and Print Module)
additional information gathered 444
control statements 438
deadlock report 439
lock states 442
overview 438
reading the report 440
reporting anomaly 444
selecting only the deadlock block 444
special situations 443
subsystem detected deadlocks 444
DFSERA40 (Program Isolation Trace Record Format and Print module)
control statements 448
output sample 445
overview 445
DFSERA40 (Program Isolation Trace Record Format and Print Module)
deadlock report 439
DFSERA50 (DL/I Call Image Capture module)
control statements 448
overview 448
DFSERA60 (IMS Trace Table Record Format and Print module)
control statements 448
overview 448
DFSERA70 (Enhanced Select exit routine)
examples 450
overview 449
DFSILTA0 (Log Transaction Analysis utility)
prerequisites 460
program inputs 460
program outputs 460
recommendations 460
requirements 460
restriction 459
DFSIST30 (Report Writer)
program modules

DFSIST30 (Report Writer) *(continued)*
program modules *(continued)*
Report Writer (DFSIST30) 473
reports produced, descriptions and examples
Line-and-Terminal report 473
Messages Queued But Not Sent (by destination) 473
Messages Queued But Not Sent (by transaction code) 473
Messages, Program-to-Program (by destination) 473
Messages, Program-to-Program (by transaction code) 473
Transaction-Response report 474
Statistical Analysis utility 473
DFSIST40 (Message Select and Copy or List)
Statistical Analysis utility 474
DFSISTS0 (Statistical Analysis utility)
examples 479
DFSISTS0 (Statistical Analysis utility)
prerequisites 471
recommendations 471
requirements 471
restrictions 471
DFSLTMG0 (Log Merge utility)
control statement format 504
controlling log merge 504
DD statements 506
input and output 503
JCL requirements 505
MSC (Multiple Systems Coupling) 503
overview 503
prerequisites 503
recommendations 503
requirements 503
restrictions 503
DFSMAP statement
keywords 134
parameters 134
DFSMARSH statement
keywords 128
parameters 128
DFSMNTR0, Data Communication Monitor 455
DFSMREC control statement 580
DFSOFMD0 (Offline Dump Formatter utility)
dump format control data set
DD statement 468
description 468
subset options 468
dump formatter 465
environments
DB batch 467
DB/DC 467
DBCTL 467
DCCTL 467
TM batch 467
input and output 466
IPCS 467
load modules 465
migration considerations 465
overview 465
prerequisites 465
recommendations 466
requirements 466
DFSOFMD0 (Offline Dump Formatter utility) *(continued)*
restrictions 465
SDUMP 467
DFSTVER0 (Time-Controlled Operations Verification utility)
EXEC statement 588
JCL specifications 588
output
description 587
error report 590
message-table report 591
statistics report 590
summary report 591
time-schedule request table 590
timer elements report 590
prerequisites 587
recommendations 587
requirements 587
restrictions 587
DFSUARC0 (Log Archive utility)
Batch DASD SLDS archive 493
control statements 497
COPY statement 499
copying log records into user data sets 493
creating an RLDS 493
DD statements 496
error processing 501
examples 502
EXIT statement 501
JCL requirements 496
OLDS input 494
omitting log records on SLDS 493
optional functions 493
overview 493
prerequisites 494
program output 495
recommendations 494
requirements 494
restrictions 494
RLDS (Recovery Log Data Set) 493
SLDS input 495
SLDS statement 498
specifying forced end of volume 493
specifying user exit routines 493
DFSUDT0x (device characteristics table)
device characteristics table 198
specifying screen size 198
DFSULTR0 (Log Recovery utility)
CLS mode 507
dual log input
CLS mode 509
DUP mode 509
REP mode 509
DUP mode 507
error block listing (SYSPRINT) 511
input 508
interim log error ID record 511
modes 510
OLDS recovery 508
overview 507
prerequisites 508
PSB mode 507
recommendations 508
REP mode 507
requirements 508

DFSULTR0 (Log Recovery utility) *(continued)*
restrictions 508
single log input 508
SLDS recovery 508
DFSUMSV0 (Multiple Systems Verification utility)
EXEC statement 570
recommendations 562
requirements 562
utility control statements 570
DFSUOCU0 (Online Change Copy utility)
active library 574
cancellation 573
DD statements 577
DFSMREC control statement 580
EXEC statement 575
inactive library 574
INITMOD procedure 579
JCL 579
libraries used 574
MSDB 573
OLCUTL procedure 578
overview 573
prerequisites 574
procedure statement 574, 580
recommendations 574
requirements 574
restrictions 573
staging library 574
DFSUOLC0 (Global Online Change utility)
examples 545
JCL 542
OLCSTAT data set 541
overview 541
parameters 543
prerequisites 542
recommendations 542
requirements 542
restrictions 542
DFSUPAA0 (MFS Language utility)
compilation statements
ALPHA statement 245
COPY statement 246
EJECT statement 250
END statement 250
EQU statement 246
PRINT statement 249
RESCAN statement 247
SPACE statement 250
STACK statement 248
TITLE statement 249
UNSTACK statement 248
control blocks 167
Format definition statements
DEV statement 189
DFLD statement 227
DIV statement 209
DO statement 224
DPAGE statement 216
ENDDO statement 239
FMT statement 188
FMTEND statement 240
PPAGE statement 223
RCD statement 226
format set 167
DFSUPAA0 (MFS Language utility) *(continued)*
Message definition statements
DO statement 182
ENDDO statement 188
LPAGE statement 179
MFLD statement 182
MSG statement 177
MSGEND statement 188
PASSWORD statement 180
SEG statement 180
modes 167
Partition set definition statements
PD statement 241
PDB statement 240
PDBEND statement 243
prerequisites 168
recommendations 168
requirements 168
restrictions 168
standard mode (MFSUTL procedure)
phase 1 252
phase 2 253
Table definition statements
IF statement 243
TABLE statement 243
TABLEEND statement 245
test mode (MFSTEST procedure)
phase 1 preprocessor 258
phase 2 258
source statement preprocessor 258
DFSUPRT0 (Spool SYSOUT Print utility)
DD statements 585
prerequisites 583
recommendations 583
requirements 583
restrictions 583
DFSURCL0 (Create RDDS from Log Records utility)
DD statements 611
description 609
examples
JCL example 614
Summary report example 614
EXEC statement 611
input and output 611
JCL specifications 611
prerequisites 610
recommendations 610
requirements 610
restrictions 610
return codes 613
Utility control statements 612
DFSURCM0 (Create RDDS from MODBLKS utility)
DD statements 621
description 619
examples
JCL example 623
Summary report example 623
EXEC statement 621
input and output 620
JCL specifications 620
prerequisites 620
recommendations 620
requirements 620
restrictions 620

DFSURCM0 (Create RDDS from MODBLKS utility) *(continued)*
return codes 622
Utility control statements 621
DFSURCP0 (Copy RDDS utility)
DD statements 606
description 605
examples
JCL example 607
Sample summary output 607
EXEC statement 606
input and output 605
JCL specifications 606
prerequisites 605
recommendations 605
requirements 605
restrictions 605
return codes 607
Utility control statements 606
DFSURDD0 (RDDS Extraction utility)
DD statements 634
description 633
examples
JCL example 636
Sample Query report 636
EXEC statement 634
input and output 633
JCL specifications 634
prerequisites 633
recommendations 633
requirements 633
restrictions 633
return codes 635
Utility control statements 634
DFSURST0 (DRD IMS SYSGEN stage 1 pre-parser utility)
DD statements 629
description 627
examples
JCL example 631
Summary report example 631
EXEC statement 629
input and output 628
JCL specifications 629
prerequisites 628
recommendations 628
requirements 628
restrictions 628
return codes 630
Utility control statements 630
DFSUSVC0 (Dynamic SVC utility)
DD statements 538
error processing 539
examples 539
input 538
JCL requirements 538
output 538
overview 537
prerequisites 537
recommendations 538
requirements 537
restrictions 537
return codes 539
DFSUTB00
prerequisites 161
recommendations 162
DFSUTB00 (MFS Device Characteristics Table utility)
DD statements 163
requirements 161
running the utility 164
DFSUTR20 (IMS Monitor Report Print utility)
analysis control data set 456
definition of terms 455
input 455
JCL example 457
JCL requirements 456
overview 455
prerequisites 455
recommendations 455
requirements 455
restrictions 455
DFSUTSA0 (MFS Service utility)
DD statements 549
EXEC statement 549
prerequisites 548
recommendations 548
requirements 548
utility control statement keywords 550
utility control statement parameters 550
DFSWTnnn procedure (Spool SYSOUT Print utility
DFSUPRT0) 584
DIF (device input format)
language statements used to create
DEV 189
DFLD 227
DIV 209
DO 223
DPAGE 216
ENDDO 239
FMT 188
FMTEND 240
PPAGE 223
RCD 226
summary 175
DIV statement
COMPR= operand 216
HDRCTL= operand 213
dividing database into multiple data set groups
DBD generation 48
DL/I Call Image Capture module (DFSERA50)
control statements 448
File Select and Formatting Print utility (DFSERA10) 448
overview 448
DOF (device output format)
language statements used to create
DEV 189
DFLD 227
DIV 209
DO 223
DPAGE 216
ENDDO 239
FMT 188
FMTEND 240
PPAGE 223
RCD 226
summary 175
DRD
utilities 593
DRD IMS SYSGEN stage 1 pre-parser utility (DFSURST0)
DD statements 629

DRD IMS SYSGEN stage 1 pre-parser utility (DFSURST0) *(continued)*
description 627
examples
JCL example 631
Summary report example 631
EXEC statement 629
input and output 628
JCL specifications 629
prerequisites 628
recommendations 628
requirements 628
restrictions 628
return codes 630
Utility control statements 630
DSPURX00 (Database Recovery Control utility)
examples 534
invoking the utility 534
prerequisites 531
recommendations 531
requirements 531
restrictions 531
dump format control data set
DD statement 468
description 468
subset options 468
dynamic resource definition (DRD)
utilities 593
Dynamic SVC utility (DFSUSVC0)
DD statements 538
error processing 539
examples 539
input 538
JCL requirements 538
output 538
overview 537
prerequisites 537
recommendations 538
requirements 537
restrictions 537
return codes 539

E

E= keyword
DFSERA10 OPTION control statement 431
EJECT statement (language utility) 250
END statement (language utility) 250
ENDDO statement
specifying to terminate
DFLD statements 239
MFLD statements 188
Enhanced Select exit routing (DFSERA70)
examples 450
overview 449
EQU statement (language utility statement) 246
equate processing 246
error block listing (SYSPRINT)
description of fields 511
examples
File Select and Formatting Print utility 432
selecting all log record types with token 451
selecting specific log record types with token 451
EXEC statement
MFS Device Characteristics Table (DFSUTB00) 162
EXEC statement *(continued)*
Spool SYSOUT Print utility (DFSUPRT0) 585
TCO Verification utility (DFSTVER0) 588
EXEC statement, operands
DEVCHAR= 170
EXIT= parameter
Data Capture exit routine 43
DBD statement 43
EXITR= keyword
DFSERA10 OPTION control statement 431

F

Fast Path
AREA statement
DEDB DBD generation record 26
description 61
format 49
keywords 61
DEDB DBD generation
description 22
examples 147
input record structure 26
DEDB PSB generation
alternate PCB statement 266
Log Analysis utility (DBFULTA0)
log intervals 401
MSDB DBD generation
description 22
examples 146
MSDB PSB generation
alternate PCB statement 266
examples 294
Fast Path DEDB
data entry database (DEDB)
DBD generation 28
Fast Path Log Analysis utility (DBFULTA0)
error processing 408
Fast Path report types 409
input and output 402
JCL requirements 403
overview 401
prerequisites 402
recommendations 402
reports
Detail-Listing-of-Exception-Transactions 409
Overall Summary of Resource Usage and Contentions 416
Overall Summary of Transit Times 416
Recapitulation-of-the-Analysis 419
Summary-of-Exception-Detail-by-Transaction-Code 415
Summary-of-Region-Occupancy 418
Summary-of-VSO-Activity 419
requirements 402
restrictions 402
utility control statements 404
Fast Path MSDB
DEDB (data entry database)
DBD generation 28
main storage database (MSDB)
DBD generation 28
FEAT= operand (DEV statement), specifying 204
FIELD statement

FIELD statement *(continued)*
DEDB database 102
description 101
format 113
GSAM database 103
HDAM and PHDAM database 104
HIDAM and PHIDAM database 106
HISAM database 107
HSAM database 108
Index database 109
keywords 113
MSDB database 110
SHISAM database 111
SHSAM database 108
field tab
specifying 201
File Select and Formatting Print utility (DFSERA10)
control statements
COMMENT 432
CONTROL 425
description 425
END 432
OPTION 427
COPY option 429
DL/I Call Image Capture module (DFSERA50) 448
Enhanced Select exit routine (DFSERA70) 449
examples 432
IMS Trace Table Record Format and Print module (DFSERA60) 448
input 423
JCL requirements
DD statements 424
description 424
examples 432, 436
NEGOF option 429
OPTION statement
PARM= parameter, subparameters of 449
optional keywords
B= 430
C= 430
COND= 430
D= 432
DDNAME= 432
E= 431
EXITR= 431
FLDLEN= 430
FLDTYP= 430
H= 430
L= 430
O= 429
OFFSET= 429
P= 432
PARM= 429
PRTSYS= 432
STARTAF= 430
STOPAFT= 430
SYM= 429
T= 430
V= 430
VALUE= 430
output 423
overview 423
prerequisites 423
PRINT option 429
File Select and Formatting Print utility (DFSERA10) *(continued)*
Program Isolation (PI) Trace Record Format and Print Module (DFSERA40)
control statements 448
overview 445
sample 445
recommendations 423
Record Format and Print Module (DFSERA30)
control statements 438
description 438
requirements 423
restrictions 423
fill characters
input message fields
specifying 187
output device fields
specifying 178
FILL= operand
MFLD statement, specifying 187
MSG statement, specifying 178
FLDLEN= keyword
DFSERA10 OPTION control statement 430
FLDTYP= keyword
DFSERA10 OPTION control statement 430
FMT statement, specifying 188
FMTCPY control statement
MFSBTCH2 procedure 257
MFSUTL procedure 254
FMTEND statement, specifying 240
forced EOV
Log Archive utility 493
format set 167
FRSPC= keyword
DATASET statement 58
FTAB= operand (DEV statement)
specifying 201

G

Generation utilities
Application Control Blocks Maintenance utility 1
Database Description (DBD) Generation utility 1
MFS Device Characteristics Table utility (DFSUTB00) 1
MFS Language utility (DFSUPAA0) 1
Program Specification Block (PSB) generation utility 1
Global Online Change utility (DFSUOLC0)
examples 545
JCL 542
OLCSTAT data set 541
overview 541
parameters 543
prerequisites 542
recommendations 542
requirements 542
restrictions 542
GSAM (Generalized Sequential Access Method)
DBD generation
example 145
specification 20, 39
PCB generation
example 292

H

H= statement
  DFSERA10 OPTION control statement 430
HALDB (High Availability Large Database)
  partitions
    database uncommitted updates restriction 263
HDAM (Hierarchical Direct Access Method)
  defining
    DFSCASE statement 136
    DFSMAP statement 133
    DFSMARSH statement 127
  DFSCASE statement 136
  DFSMAP statement 133
  DFSMARSH statement 127
HDAM database
  DBD (Database Description) generation 28
HDRCTL= operand
  DIV statement, specifying 213
HIDAM (Hierarchical Indexed Direct Access Method)
  defining
    DFSCASE statement 136
    DFSMAP statement 133
    DFSMARSH statement 127
  DFSCASE statement 136
  DFSMAP statement 133
  DFSMARSH statement 127
HIDAM database
  DBD (Database Description) generation 28
High Availability Large Database (HALDB)
  partitions
    database uncommitted updates restriction 263
HISAM (Hierarchical Indexed Sequential Access Method)
  defining
    DFSCASE statement 136
    DFSMAP statement 133
    DFSMARSH statement 127
  DFSCASE statement 136
  DFSMAP statement 133
  DFSMARSH statement 127
HISAM database
  DBD (Database Description) generation 28
HSAM (Hierarchical Sequential Access Method)
  defining
    DFSCASE statement 136
    DFSMAP statement 133
    DFSMARSH statement 127
  DFSCASE statement 136
  DFSMAP statement 133
  DFSMARSH statement 127
HSAM database
  DBD (Database Description) generation 28

I

IF statement
  specifying 243
IMS catalog
  ACB Generation and Catalog Populate utility (DFS3UACB) 313
  ACB library
    copying 331, 339
  ACBGEN and population in a single step 313
  copying 331, 339
IMS catalog *(continued)*
  DBD library
    copying 331, 339
  DBRC, without 329, 369
  defining 329, 369
  loading 373
  loading during ACBGEN 313
  maintenance 365
  populating 373
  populating during ACBGEN 313
  PSB library
    copying 331, 339
  purging records 389
  segments, deleting 389
  sizing 385
  statistics 385
  utilities
    DFS3UCD0 329, 369
IMS Catalog Copy utilities (DFS3CCE0, DFS3CCI0)
  overview 331
IMS Catalog Export utility (DFS3CCE0)
  DD statements 336
  input 332
  JCL 334
  output 332
  overview 331
  prerequisites 332
  recommendations 332
  requirements 332
  restrictions 332
  return codes 339
  statistics report, export 337
IMS Catalog Import utility (DFS3CCI0
  input 340
  JCL 343
  output 340
  prerequisites 340
  recommendations 340
  requirements 340
  restrictions 340
  return codes 352
  statistics report, import 350
IMS Catalog Import utility (DFS3CCI0)
  DD statements
    SYSINP 345
  JCL 343
  overview 339
  SYSINP DD statement 345
IMS Catalog Library Builder utility (DFS3LU00) 357
IMS Catalog Maintenance utility (DFS3CM00)
  DD statements 366
  input 365
  output 365
  overview 365
  requirements 365
  restrictions 365
  return codes 367
IMS Catalog Populate utility (DFS3PU00)
  DD statements
    MANAGEDACBS statement, SYSINP 324, 347, 382
    SYSINP 318, 380
  input 375
  JCL 376
  output 375

IMS Catalog Populate utility (DFS3PU00) *(continued)*
overview 373
prerequisites 374
recommendations 375
requirements 374
restrictions 374
return codes 387
statistics report 385
SYSINP DD statement
MANAGEDACBS statement 324, 347, 382
IMS Catalog Record Purge utility (DFS3PU10) 389
IMS catalog utilities 311
IMS Dump Formatter 467
IMS Monitor Report Print utility (DFSUTR20)
analysis control data set 456
definition of terms 455
input 455
JCL example 457
JCL requirements 456
overview 455
prerequisites 455
recommendations 455
requirements 455
restrictions 455
IMS Monitor Reports
DB/DC
Log Merge utility (DFSLTMG0) 503
Log Recovery utility (DFSULTR0) 507
Log Transaction Analysis utility (DFSILTA0) 459
Statistical Analysis utility (DFSISTS0) 471
Log Archive utility (DFSUARC0) 493
Offline Dump Formatter utility (DFSOFMD0) 465
IMS password
PASSWORD statement 180
specifying 230
IMS Trace Table Record Format and Print module (DFSERA60)
control statements 448
File Select and Formatting Print utility (DFSERA10) 448
overview 448
IMS-issued subsystem detected deadlocks 444
IMS.FORMAT library
backup and restore operations 259, 260
IMS.REFERAL library
backup and restore operations 259, 260
IMSMSV procedure (Multiple Systems Verification utility DFSUMSV0) 569
IMSWTnnn procedure (Spool SYSOUT Print utility DFSUPRT0) 586
index database
DBD (Database Description) generation 28
Index database
defining
DFSCASE statement 136
DFSMAP statement 133
DFSMARSH statement 127
DFSCASE statement 136
DFSMAP statement 133
DFSMARSH statement 127
INDEX DBD generation
logical DBD 24
overview 23
primary HIDAM index 23
secondary index 23
INDEX function (MFS Service utility DFSUTSA0) 553
INITMOD procedure
DFSMREC control statement 580
MODSTAT record 580
procedure statement 580
IPCS (Interactive Problem Control System)
IMS Dump Formatter 467
Offline Dump Formatter 467
Offline Dump Formatter, user control statement 467
iterative processing (MFLD/DFLD)
DO statement 183, 224
ENDDO statement 188, 239
PRINT GEN effects 249
RCD statement with DFLD 226

J

JUST= operand (MFLD statement), specifying 186
justification
specifying 186

K

keyboard shortcuts ix

L

L= keyword
DFSERA10 OPTION control statement 430
LABEL field
DBD generation 48
LCHILD statement
HDAM databases 95
HIDAM databases 96
HISAM databases 94
INDEX databases 97
PHDAM databases 96
PHIDAM databases 97
PSINDEX databases 98
legal notices
notices 639
trademarks 639, 640
LIST function (MFS Service utility DFSUTSA0) 557
literal fields
output message
length, password parameter 230
specifying length 185
truncating literals 225
Log Analysis utility
Fast Path (DBFULTA0) 401
Log Archive utility (DFSUARC0)
Batch DASD SLDS archive 493
control statements 497
COPY statement 499
copying log records into user data sets 493
creating an RLDS 493
DD statements 496
error processing 501
examples 502
EXIT statement 501
JCL requirements 496
OLDS input 494
omitting log records on SLDS 493

Log Archive utility (DFSUARC0) *(continued)*
optional functions 493
overview 493
prerequisites 494
program output 495
recommendations 494
requirements 494
restrictions 494
RLDS (Recovery Log Data Set) 493
SLDS input 495
SLDS statement 498
specifying forced end of volume 493
specifying user exit routines 493
Log Merge utility (DFSLTMG0)
control statement format 504
controlling log merge 504
DD statements 506
input and output 503
JCL requirements 505
MSC (Multiple Systems Coupling) 503
overview 503
prerequisites 503
recommendations 503
requirements 503
restrictions 503
LOG parameter
DBD statement 45
log record
Statistical Analysis utility 471
Log Recovery utility (DFSULTR0)
active region messages 513
CLS mode 507
CLS mode error listing 511
control statements 516
creating a new log 518
creating an interim log 517
DD statements 515
dual log input
CLS mode 509
DUP mode 509
REP mode 509
Dump of data records 513
DUP mode 507
DUP mode error listing 511
error block listing (SYSPRINT) 511
error processing 519
examples 519
input 508
interim log error ID record 511
JCL requirements 514
modes 507, 510
OLDS recovery 508
output 510
overview 507
prerequisites 508
print active PSB reports 518
PSB mode 507
recommendations 508
REP mode 507
REP mode verification messages 513
requirements 508
restrictions 508
single log input 508
SLDS recovery 508
Log Transaction Analysis utility (DFSILTA0)
description 459
parameter descriptions 460
prerequisites 460
program inputs 460
program outputs 460
recommendations 460
requirements 460
restriction 459
Log utilities
Log Archive utility (DFSUARC0) 491
Log Merge utility (DFSLTMG0) 491
Log Recovery utility (DFSULTR0) 491
logical
links
multisystem control block 563
terminals
multisystem control block 564
LOGICAL parameter
DATASET statement 52
logical terminals, multisystem control block 561
LPAGE
operands
COND= 179
SOR= 179
output
conditional selection 179
LTH= operand (MFLD statement), specifying 185
LUSIZE= operand (PDB statement), specifying 240

M

main storage database (MSDB)
DBD generation 28
making changes online
Global Online Change Copy utility 541
Online Change Copy utility 573
MANAGEDACBS statement 324, 347, 382
marshal attributes 127
MBR=parameter
DBD generation 158
metadata
application
ACB Generation and Catalog Populate utility (DFS3UACB) 313
loading into the IMS catalog 373
database
ACB Generation and Catalog Populate utility (DFS3UACB) 313
loading into the IMS catalog 373
MFLD (message field statement)
ATTR= operand 186
FILL= operand 187
iterative processing 182, 183
JUST= operand 186
LTH= operand 185
printing generated MFLD statements 187
MFS Device Characteristics Table utility (DFSUTB00)
DD statements 163
description 161
EXEC statement
description 162
MFS descriptor format 161
MFSDCT procedure 162

MFS Device Characteristics Table utility (DFSUTB00) *(continued)*
prerequisites 161
PROC statement
description 162
recommendations 162
requirements 161
restrictions 161
running the utility 164
MFS Language utility
compilation statements
ALPHA 172
COPY 245
EJECT 250
END 250
EQU 246
RESCAN 247
SPACE 250
summary 176
SYSIN 172
SYSLIB 172
SYSPRINT 172
TITLE 249
UNSTACK 248
concatenated equates 247
MFS Language utility (DFSUPAA0)
batch mode
description 254
MFSBTCH1 procedure 255
MFSBTCH2 procedure 256
compilation statements
ALPHA statement 245
COPY statement 246
EJECT statement 250
END statement 250
EQU statement 246
PRINT statement 249
RESCAN statement 247
SPACE statement 250
STACK statement 248
TITLE statement 249
UNSTACK statement 248
control blocks 167
ddnames (MFSRVC)
FORMAT 172
REFIN 172
SYSIN 172
SYSPRINT 172
SYSSNAP 172
ddnames (MFSULT, MFSBTCH1, and MFSBTCH2)
DUMMY 171
FORMAT 171
REFIN 171
REFOUT 171
REFRD 171
SYSIN 171
SYSLIB 171
SYSUT3 171
SYSUT4 171
UTPRINT 172
description 167
FMTCPY control statement
MFSBTCH2 procedure 257
MFSUTL procedure 254
Format definition statements
MFS Language utility (DFSUPAA0) *(continued)*
Format definition statements *(continued)*
DEV statement 189
DFLD statement 227
DIV statement 209
DO statement 224
DPAGE statement 216
ENDDO statement 239
FMT statement 188
FMTEND statement 240
PPAGE statement 223
RCD statement 226
format set 167
JCL parameter descriptions
PCOMP= 168
PSUBS= 168
PXREF= 168
JCL requirements
MFSBACK procedure 259
MFSBTCH1 procedure 255
MFSBTCH2 procedure 256
MFSREST procedure 260
MFSTEST procedure 258
MFSUTL procedure 253
Message definition statements
DO statement 182
ENDDO statement 188
LPAGE statement 179
MFLD statement 182
MSG statement 177
MSGEND statement 188
PASSWORD statement 180
SEG statement 180
modes 167
Partition set definition statements
PD statement 241
PDB statement 240
PDBEND statement 243
prerequisites 168
recommendations 168
REFCPY control statement
MFSBTCH1 procedure 256
MFSULT procedure 254
requirements 168
restrictions 168
standard mode (MFSUTL procedure)
phase 1 252
phase 2 253
region parameter estimate 171
Table definition statements
IF statement 243
TABLE statement 243
TABLEEND statement 245
test mode (MFSTEST procedure)
description 257
phase 1 preprocessor 258
phase 2 258
region parameter estimate 171
source statement preprocessor 258
MFS Service utility (DFSUTSA0)
DD statements 549
DELETE function 553
description 547
EXEC statement 549

MFS Service utility (DFSUTSA0) *(continued)*
INDEX function 553
LIST function
output 559
LIST function output 557
MFSRVC procedure 548
prerequisites 548
PROC statement
description 548
recommendations 548
RELATE function 556
requirements 548
restrictions 547
SCRATCH function 554
utility control statement keywords
FMT= 550
utility control statement parameters 550
utility control statements 549
MFSBACK procedure (MFS Language utility)
description 259
JCL requirements 259
MFSBTCH1 procedure (MFS Language utility)
description 255
JCL requirements 255
MFSBTCH2 procedure (MFS Language utility)
description 256
JCL requirements 256
MFSDCT
MFS Device Characteristics Table utility (DFSUTB00) 161
MFSREST procedure (MFS Language utility)
description 260
JCL requirements 260
MFSRVC procedure (MFS Service utility) 548
MFSTEST procedure (MFS Language utility)
JCL requirements 258
region parameter estimate 171
step 1 (phase 1) 258
step 1 (source statement preprocessor) 258
step 2 (phase 2) 258
MFSUTL procedure (MFS Language utility)
JCL requirements 253
region parameter estimate 171
step 1 (phase 1) 252
step 1 (preprocessor) 252
step 2 (phase 2) 253
MID (message input descriptor)
language statements used to create
DO 182
ENDDO 188
LPAGE 179
MFLD 182
MSG 177
MSGEND 188
PASSWORD 180
SEG 180
summary 175
MOD (message output descriptor)
language statements used to create
DO 182
ENDDO 188
LPAGE 179
MFLD 182
MSG 177
MOD (message output descriptor) *(continued)*
language statements used to create *(continued)*
MSGEND 188
PASSWORD 180
SEG 180
summary 175
MODEL= parameter
DATASET statement 52
modified data tag (MDT) 233
MODSTAT record, INITMOD procedure 580
MSC (Multiple Systems Coupling)
Log Merge utility
input 503
output 503
MSDB (Main Storage Database)
defining
DFSCASE statement 136
DFSMAP statement 133
DFSMARSH statement 127
DFSCASE statement 136
DFSMAP statement 133
DFSMARSH statement 127
MSDB DBD generation
description 22
MSG statement
FILL= operand 178
MSGEND statement
specifying 188
Multiple Systems Verification utility (DFSUMSV0)
description 561
EXEC statement 570
IMSMSV procedure 569
input validation 562
invoking the procedure 570
logical links 563
logical terminals 561, 564
MSC (Multiple Systems Coupling) 561
multisystem control block verification 562
multisystem path map 564
output messages 564
partner IDs 562
physical links 563
prerequisites 561
PROC statement
description 570
procedure for executing 569
processing phases 562
recommendations 562
requirements 562
restrictions 561
SYSID paths
description 563
local 563
remote 563
transaction code attributes
consistency between systems 563
description 563
local 563
remote 563
utility control statements
description 570
verification process 562
multisystem control blocks 562
multisystem path map, MSC 564

N

NAME parameter
  DBD statement 39
NEGOF option
  File Select and Formatting Program (DFSERA10) 429
NOLOG parameter
  DBD statement 45
not message-driven option
  Fast Path Log Analysis utility 406
null
  compression, specifying 216
  fill character
    input message fields 187
    output device fields 178

O

O= keyword
  control statements
    DFSERA10 CONTROL 427
    DFSERA10 OPTION 429
Offline Dump Formatter utility (DFSOFMD0)
  dump format control data set
    DD statement 468
    description 468
    subset options 468
  dump formatter 465
  environments
    DB batch 467
    DB/DC 467
    DBCTL 467
    DCCTL 467
    TM batch 467
  input and output 466
  IPCS 467
  load modules 465
  migration considerations 465
  overview 465
  prerequisites 465
  recommendations 466
  requirements 466
  restrictions 465
  SDUMP 467
OFFSET= keyword
  DFSERA10 OPTION control statement 429
OLCSTAT data set
  description 541
  initializing 541
  recover procedure 541
OLCUTL procedure, process 578
OLDS (online log data set)
  dual OLDSs 494
  input to Log Archive utility 494
  recover points 494
  recovery using the Log Recovery utility 508
  termination 494
OM Audit Trail Format and Print module (CSLULALE)
  DD statements 452
  description 451
  JCL specifications 451
  Limiting log data to a specified time range 452
  Utility control statements 452
omitting log records on SLDS
omitting log records on SLDS *(continued)*
  Log Archive utility 493
Online Change Copy utility (DFSUOCU0)
  active library 574
  cancellation 573
  DD statements 577
  DFSMREC control statement 580
  EXEC statement 575
  inactive library 574
  INITMOD procedure 579
  JCL 579
  libraries used 574
  MSDB 573
  OLCUTL procedure 578
  overview 573
  prerequisites 574
  procedure statement 574, 580
  recommendations 574
  requirements 574
  restrictions 573
  staging library 574
Online Database Image Copy utility (DFSUICP0)
  PSBGEN specifications required 309
operator control tables
  language statements used to create
    IF 243
    TABLE 243
    TABLEEND 245
OSAM data sets block size 56
output message
  header
    structure and content 213
Overall Summary of Resource Usage and Contentions for All Transaction Codes and PSBs Report
  Fast Path Log Analysis utility 416
Overall Summary of Transit Times by Transaction Code for IFP Regions Report
  Fast Path Log Analysis utility 416
OVFLW= parameter
  DATASET statement 53

P

P= keyword
  DFSERA10 OPTION control statement 432
PARM= keyword
  DFSERA10 OPTION control statement 429
  subparameters of
    TOKEN= subparameter 450
    XFMT= subparameter 449
partition set, language statements used to create
  PD 241
  PDB 240
  PDBEND 242
Partitioned Hierarchical Direct Access Method (PHDAM) 145
Partitioned Hierarchical Indexed Direct Access Method (PHIDAM) 145
partitions
  database uncommitted updates restriction 263
PASSWD= parameter
  DBD statement 42
PASSWORD parameter (DFLD statement), specifying 230
PASSWORD statement, specifying 180
password, IMS

password, IMS *(continued)*
specifying 230
PCB statement
database PCB size 269
Full-function or Fast Path database 269
GSAM 281
SENSEG statement 283
PCBs (program control blocks)
DB
database uncommitted updates restriction 263
PD statement (partition definition)
specifying 241
PDB (partition descriptor block)
language statements used to create
PDBEND 176
summary 176
LUSIZE= operand 240
PDBEND statement, specifying 242
PFK= operand (DEV statement), specifying 205
PHDAM (Partitioned Hierarchical Direct Access Method)
DBD generation
example 145
defining
DFSCASE statement 136
DFSMAP statement 133
DFSMARSH statement 127
DFSCASE statement 136
DFSMAP statement 133
DFSMARSH statement 127
PHIDAM (Partitioned Hierarchical Indexed Direct Access Method)
DBD generation
example 145
defining
DFSCASE statement 136
DFSMAP statement 133
DFSMARSH statement 127
DFSCASE statement 136
DFSMAP statement 133
DFSMARSH statement 127
physical links, MSC 563
PPAGE statement, specifying 223
PRINT option
File Select and Formatting Print utility (DFSERA10) 429
PRINT statement (language utility) 249
printed page format control
bottom margin 207
line density 208
top margin 207
PROC statement
MFS Device Characteristics Table utility (DFSUTB00) 162
Multiple Systems Verification utility (DFSUMSV0) 570
Spool SYSOUT Print utility (DFSUPRT0) 584
procedure
ACBGEN 5
DBDGEN 157
INITMOD 579
OLCUTL 578
PSBGEN 308
program control blocks (PCBs)
DB
database uncommitted updates restriction 263
program function keys (3270)
program function keys (3270) *(continued)*
specifying 205
Program Isolation (PI) Trace Record Format and Print Module (DFSERA40)
control statements 448
output sample 445
Program Isolation Trace Record Format and Print module (DFSERA40)
overview 445
Program Isolation Trace Record Format and Print Module (DFSERA40)
deadlock report 439
program output
Log Archive utility 495
Program Specification Block (PSB) Generation utility
Fast Path databases
processing options 278
program tab function
fill character 178
protecting the screen
specifying parameter on DLFD statement 232
PRTSYS= keyword
DFSERA10 OPTION 432
PSB (Program Specification Block)
control statement formats
alternate PCB 266
END 291
Full-function or Fast Path database PCB 269
GSAM PCB 281
I/O PCB 266
PSBGEN 287
SENFLD 286
SENSEG 283
description 263
dynamic option
adding in IMS-managed ACB environments 13
examples
application database 299
Fast Path 294
Field Level Sensitivity 293
GSAM 292
logical database 295
sample hierarchic data structure 291
shared secondary index 304
execution 309
Full-function or Fast Path database PCB
PROCOPT parameter 269
generating
control statement formats 12, 13
control statements 266
overview 263
PCBNAME= parameter 293
PCBs (Program Communication Blocks) 263
specifying options for DEDBs
END statement 291
PSBGEN statement 287
SENSEG statement 283
PSB Generation utility
Fast Path databases
processing options 278
PSB library
copying, with IMS catalog 331, 339
PSB= keyword

PSB= keyword *(continued)*
ACB Generation and Catalog Populate utility (DFS3UACB) 320
PSB= parameter
ACB Maintenance utility 8
PSBGEN
procedure 308
statement
maximum number of database PCBs 269
PSB generation 287
PSBGEN utility
generating
input and output 264
prerequisites 263
recommendations 264, 308
requirements 264
restrictions 263
Rules 264
six input/output statement types 264
PT (program tab) function
fill character 178

R

RCD statement, specifying 226
RDDS Extraction utility (DFSURDD0)
DD statements 634
description 633
examples
JCL example 636
Sample Query report 636
EXEC statement 634
input and output 633
JCL specifications 634
prerequisites 633
recommendations 633
requirements 633
restrictions 633
return codes 635
Utility control statements 634
RDDS to Repository utility (CSLURP10)
DD statements 600
description 599
examples
JCL example 602
Sample summary output 602
EXEC statement 600
input and output 600
JCL specifications 600
prerequisites 599
recommendations 600
requirements 599
restrictions 599
return codes 601
utility control statements 601
Recapitulation-of-the-Analysis
Report
Fast Path Log Analysis utility 419
RECFM= parameter
DATASET statement 58
RECON data set
upgrading 531
version migration 531
Record Format and Print Module (DFSERA30)
control statements 438
deadlock report 439
overview 438
Record Format and Print Module (DFSERS30)
additional information gathered 444
Deadlock report 439
File Select and Formatting Print utility (DFSERA10) 438
lock states 442
reading the report 440
reporting anomaly 444
resultant state of the lock 442
selecting only the deadlock block 444
special situations 443
subsystem detected deadlocks 444
RECORD= parameter
DATASET statement 57
REFCPY control statement
MFSBTCH1 procedure 256
MFSULT procedure 254
REL= parameter
DATASET statement 59
RELATE function (MFS Service utility DFSUTSA0) 556
Repository to RDDS utility (CSLURP20)
DD statements 596
description 595
examples
JCL example 597
Sample summary output 598
EXEC statement 596
input and output 595
JCL specifications 596
prerequisites 595
recommendations 595
requirements 595
restrictions 595
return codes 597
Utility control statements 597
RESCAN statement (language utility) 247
restore operations, MFS library 260
RGN= parameter
procedures
ACBGEN 6
DBDGEN 158
RLDS (Recovery Log Data Set)
creating 493
output to Log Archive utility 495
RMNAME= parameter
DBD statement 41
ROOT= parameter
AREA statement 62

S

SCA (system control area)
specifying 184
SCAN= parameter
DATASET statement 58
SCRATCH function (MFS Service utility DFSUTSA0) 554
screen formatting
specifying screen size 198
script member, error 587
SDUMP
Offline Dump Formatter 467

SEARCHA= parameter
DATASET statement 58
secondary index
DBD generation 23
SEG statement
EXIT= operand 180
GRAPHIC= operand 180
SEGM statement
database
DEDB 63
GSAM 65
HDAM 65
HIDAM 67
HISAM 69
HSAM 71
INDEX 72
MSDB 73
PHDAM 73
PHIDAM 75
PSINDEX 77
SHSAM 78
description 63
format 78
keyword abbreviations 78
pointer keyword options and abbreviations 81
selecting
extended log formatting for X'50' log records
XFMT= subparameter 449
log records by recovery token
example of selecting all record types 451
example of selecting specific record types 451
TOKEN= subparameter 450
selector pen, 3270
specifying field detectability 232
SENSEG statement
PSB generation 283
Service utilities
Batch SPOC utility (CSLUSPOC) 525
Database Recovery Control utility (DSPURX00) 525
Dynamic SVC utility (DFSUSVC0) 525
Global Online Change utility (DFSUOLC0) 525
MFS Service utility (DFSUTSA0) 525
Multiple Systems Verification utility (DFSUMSV0) 525
Online Change Copy utility (DFSUOCU0) 525
Resource Definition Data Set Extraction utility (DFSURDD0) 525
Spool SYSOUT Print utility (DFSUPRT0) 525
Time-Controlled Operations Verification utility (DFSTVER0) 525
SHISAM (Simple Hierarchical Indexed Sequential Access Method)
defining
DFSCASE statement 136
DFSMAP statement 133
DFSMARSH statement 127
DFSCASE statement 136
DFSMAP statement 133
DFSMARSH statement 127
SHSAM (Simple Hierarchical Sequential Access Method)
defining
DFSCASE statement 136
DFSMAP statement 133
DFSMARSH statement 127
DFSCASE statement 136
SHSAM (Simple Hierarchical Sequential Access Method) *(continued)*
DFSMAP statement 133
DFSMARSH statement 127
single log input
Log Recovery utility 508
SIZE= parameter
AREA statement 61
DATASET statement 55
SKIP= parameter
File Select and Formatting Print utility (DFSERA10) 426
SLDI= operand (DEV statement), specifying 208
SLDP= operand (DEV statement), specifying 208
SLDS (system log data set)
batch archive 493
input to Log Archive utility 495
omitting log records on 493
output to Log Archive utility 495
recovery using the Log Recovery utility 508
SLDS (system log data set) statement
Log Archive utility 498
SLU
type 2, defining to operate with MFS
copy function 232
SOR= operand (LPAGE statement), specifying 179
SOUT= parameter
procedures
ACBGEN 5
DBDGEN 158
SPACE statement (language utility) 250
Specifying DBD generation
Fast Path secondary index database 23
full-function secondary index database 23
GSAM database 20
HIDAM and PHIDAM database 21
HISAM database 20
HSAM database 19, 22
Index and PSINDEX databases 23
Logical database 24
MSDB database
header information (BHDR) 22
SHISAM database 20
SHSAM database 19
Spool SYSOUT Print utility (DFSUPRT0)
blocking factors 583
DD statements 585
description 583
DFSWTnnn procedure 584
EXEC statement 585
IMSWTnnn procedure 586
prerequisites 583
PROC statement
description 584
recommendations 583
requirements 583
restrictions 583
sample output 583
system messages 583
STACK statement (MFS Language utility) 248
STARTAF= statement
DFSERA10 OPTION control statement 430
Statistical Analysis utility (DFSISTS0)
control statements 477
description 471
examples 479

Statistical Analysis utility (DFSISTS0) *(continued)*
input 471
JCL requirements
description 475
job-stream example 479
output 471
overview 471
prerequisites 471
program modules
Message Select and Copy or List (DFSIST40) 474
recommendations 471
reports produced, descriptions and examples
Application-Accounting report 474, 486
IMS-Accounting report 474, 486
Line-and-Terminal report 481
messages produced by Message Select and Copy (DF) 477, 488
Messages Queued But Not Sent (by destination) 473, 480
Messages Queued But Not Sent (by transaction code) 482
Messages, Program-to-Program (by destination) 480
Messages, Program-to-Program (by transaction code) 483
Transaction report 485
requirements 471
restrictions 471
utility control statements
descriptions 477
nonprintable character 478
symbolic terminal name 477
time 478
transaction code 477
STOPAFT= statement
DFSERA10 OPTION control statement 430
File Select and Formatting Print utility (DFSERA10) 426
SUB= operand (DEV statement)
specifying 208
Summary-of-Exception-Detail-by-Transaction-Code (for IFP Regions) Report
Fast Path Log Analysis utility 415
Summary-of-Region-Occupancy Report
Fast Path Log Analysis utility 418
Summary-of-VSO-Activity Report
Fast Path Log Analysis utility 419
SVC utility 537
SYM= option
DFSERA10 OPTION control statement 429
syntax diagram
how to read vii
SYS2= parameter
procedures
ACBGEN 6
DBDGEN 158
SYSID paths, MSC 563
SYSIN/SYSLIB record stacking and unstacking
STACK statement 248
UNSTACK 248
SYSINP DD statement 318, 345, 380
SYSMDUMP
Offline Dump Formatter 467
SYSPRINT
SYSPRINT *(continued)*
Log Archive utility 495
SYSPRINT listing control
compilation statements 172
EJECT statement 250
PRINT statement 249
SPACE statement 250
TITLE statement 249
system checkpoints
database uncommitted updates restriction 263
system literals
other formats, CA parameter (MFLD statement) 184

T

T= keyword
DFSERA10 OPTION control statement 430
tabbing
field tabs 201
TABLE statement, specifying 243
TABLEEND statement, specifying 245
TCO Error Report
Time-Controlled Operations Verification utility (DFSTVER0) 590
TCO script library
Time-Controlled Operations Verification utility (DFSTVER0) 587
TCO Time-Schedule Request Table
Time-Controlled Operations Verification utility (DFSTVER0) 590
TCO Verification procedure
Time-Controlled Operations Verification utility (DFSTVER0) 588
TCO Verification utility (DFSTVER0)
DD statements 588
TCO-Message-Table Report
Time-Controlled Operations Verification utility (DFSTVER0) 591
TCO-Statistics Report
Time-Controlled Operations Verification utility (DFSTVER0) 590
TCO-Summary Report
Time-Controlled Operations Verification utility (DFSTVER0) 591
TCO-Timer-Elements Report
Time-Controlled Operations Verification utility (DFSTVER0) 590
Time-Controlled Operations Verification utility (DFSTVER0)
description 587
EXEC statement 588
JCL specifications 588
output
description 587
error report 590
message-table report 591
statistics report 590
summary report 591
timer elements report 590
prerequisites 587
recommendations 587
requirements 587
restrictions 587
return codes 588
time-schedule request table

time-schedule request table *(continued)*
Time-Controlled Operations Verification utility (DFSTVER0) 590
TITLE statement (language utility) 249
TOKEN= subparameter 450
trademarks 639, 640
transaction code attributes (MSC) 563
Transaction-Response report
Statistical Analysis utility (DFSISTS0)
reports 474
translation, character
input messages specifying 208

U

unprotecting the screen
specifying parameter on DLFD statement 232
UNSTACK statement (language utility) 248
UOW= parameter
AREA statement 62
utilities
Batch SPOC utility (CSLUSPOC) 527
catalog Directory Recovery utility (DFS3RU00) 353
Copy RDDS utility (DFSURCP0) 605
Create RDDS from Log Records utility (DFSURCL0) 609, 614
Create RDDS from MODBLKS utility (DFSURCM0) 619
CSLULALE (OM Audit Trail Format and Print module) 451
CSLURP10 (RDDS to Repository utility) 599
CSLURP20 (Repository to RDDS utility) 595
CSLUSPOC (Batch SPOC utility) 527
DBDGEN
control statements 15
databases used with 19
information specified in 15
DFS3UCD0 329, 369
DFSURCL0 (Create RDDS from Log Records utility) 609
DFSURCM0 (Create RDDS from MODBLKS utility) 619
DFSURCP0 (Copy RDDS utility) 605
DFSURDD0 (RDDS Extraction utility) 633
DFSURST0 (DRD IMS SYSGEN stage 1 pre-parser utility) 627
DRD IMS SYSGEN stage 1 pre-parser utility (DFSURST0) 627
Dynamic SVC utility (DFSUSVC0) 537
Fast Path Log Analysis utility (DBFULTA0) 401
File Select and Formatting Print utility (DFSERA10) 423
Global Online Change utility (DFSUOLC0) 541
IMS Monitor Report Print utility (DFSUTR20) 455
Log Archive utility (DFSUARC0) 493
Log Merge utility (DFSLTMG0) 503
Log Recovery utility (DFSULTR0) 507
Log Transaction Analysis utility (DFSILTA0) 459
MFS Device Characteristics Table utility (DFSUTB00) 161
MFS Language utility (DFSUPAA0) 167
MFS Service utility (DFSUTSA0) 547
Multiple Systems Verification utility (DFSUMSV0) 561
Offline Dump Formatter utility (DFSOFMD0) 465
OM Audit Trail Format and Print module (CSLULALE) 451
Online Change Copy utility (DFSUOCU0) 573
PSBGEN 263
RDDS Extraction utility (DFSURDD0) 633
RDDS to Repository utility (CSLURP10) 599
utilities *(continued)*
Repository to RDDS utility (CSLURP20) 595
Spool SYSOUT Print utility (DFSUPRT0) 583
Statistical Analysis utility (DFSISTS0) 471
Time-Controlled Operations Verification utility (DFSTVER0) 587

V

V= parameter
DFSERA10 OPTION control statement 430
VALUE= parameter
DFSERA10 OPTION control statement 430
verification process
Multiple Systems Verification utility (DFSUMSV0) 562
VERSID= operand (DEV statement), specifying 208
version identification
specifying 208
VERSION parameter
DBD statement 46
VTAB= operand (DEV statement)
specifying 207

W

WADS (write-ahead data set)
CLS mode 507, 513
data set 515
NOWADS 517

X

XDFLD statement
description 123
format 123
HDAM database 124
HISAM database 123
keywords 123
parameter description 125
PHDAM database 124
XFMT= subparameter 449

Product Number: 5635-A06
5655-DS5
5655-TM4