Contents

Figures. ................................................................. ix
Tables. ................................................................. xiii

About This Book. ....................................................... xv
Organization of This Book ........................................ xv
Prerequisite Knowledge ............................................ xv
Organization of Utility Descriptions .......................... xvi
How to Read Syntax Diagrams .................................... xvi
CICS, DBCTL, and DCCTL ........................................... xviii
How to Send Your Comments ........................................ xviii

Summary of Changes .................................................. xxi
Changes to the Current Edition of This Book for IMS Version 8 .................................................. xxi
Changes to This Book for IMS Version 8 ........................ xxi
Library Changes for IMS Version 8 ............................... xxi

Part 1. Generation Utilities ........................................... 1

Chapter 1. Database Description (DBD) Generation .......... 3
  Information Specified in DBD Generation ..................... 4
  DBD Generation for Database Types ............................ 4
  DBDGEN Procedure ................................................... 11
  DBDGEN Statements .................................................. 13
  Output for DBD Generation ........................................ 85
  Examples of DBD Generation ....................................... 89

Chapter 2. Program Specification Block (PSB) Generation .... 111
  Input and Output of PSB Generation ............................. 111
  PSBGEN Procedure ................................................... 113
  Utility Control Statements for PSB Generation .............. 115
  Output Messages and Statistics for PSB Generation ....... 136
  Examples of PSB Generation ........................................ 137

Chapter 3. Application Control Blocks Maintenance Utility ...... 155
  Restrictions for ACBGEN. ........................................... 156
  Input and Output for ACBGEN. ..................................... 156
  Utility Control Statements for ACBGEN ......................... 159
  Error Processing for ACBGEN ...................................... 163
  Examples of ACBGEN. ............................................... 163

Part 2. Service Utilities ............................................. 165

Chapter 4. Dynamic Allocation Macro (DFSMDA) ................. 167
  Restrictions for DFSMDA ........................................... 169
  Input and Output for DFSMDA ..................................... 169
  IMSDALOC Procedure ............................................... 170
  Macro Statements for DFSMDA .................................... 172
  Examples of DFSMDA ................................................ 177

Chapter 5. Security Maintenance Utility (DFSISMP0) .......... 179
  Input and Output Flow for DFSISMP0 ............................ 180
Restrictions for DFSISMPO ................................. 181
Security Options for DFSISMPO .......................... 181
Resource Access Control Facility (RACF). ............... 183
Command Authorization Exit Routine ..................... 183
IMS Application Resource Access Security ............... 183
SECURITY Procedure .................................... 184
Utility Control Statements for DFSISMPO. ............... 188
Output for DFSISMPO. ................................ 190
Examples of DFSISMPO .................................. 191

Chapter 6. Online Change Utilities and Procedures ........ 197
Online Change Copy Utility (DFSUOCU0) ................... 197
Global Online Change Utility (DFSUOLC0) ................ 204
Global Online Change Utility Examples ................... 207

Chapter 7. Dynamic SVC Utility (DFSUSVC0) ................. 209
Restrictions for DFSUSVC0 ................................ 209
Input and Output for DFSUSVC0 ......................... 209
Error Processing for DFSUSVC0 ......................... 210
JCL Requirements for DFSUSVC0 ......................... 210
Example of DFSUSVC0 .................................. 211

Part 3. Log Utilities ......................................... 213

Chapter 8. Log Archive Utility (DFSUARC0) ................. 215
OLDS Archive .............................................. 215
Batch DASD Log Data Set Archive ......................... 216
Optional Functions for DFSUARC0 ......................... 216
Utility Input for DFSUARC0 ............................... 217
Utility Output for DFSUARC0 .............................. 218
JCL Requirements for DFSUARC0 ......................... 220
Utility Control Statements for DFSUARC0 ................. 222
Error Processing for DFSUARC0 ......................... 225
Examples of DFSUARC0 .................................. 226

Chapter 9. Log Merge Utility (DFSLTMG0) ..................... 229
Restrictions for DFSLTMG0 ............................... 229
Input and Output for DFSLTMG0 ......................... 229
JCL Requirements for DFSLTMG0 ......................... 231

Chapter 10. Log Recovery Utility (DFSULTR0) ................. 233
OLDS Recovery ............................................. 234
SLDS Recovery ............................................. 234
Input for DFSULTR0 ...................................... 234
Output for DFSULTR0 .................................... 236
JCL Requirements for DFSULTR0 ......................... 241
Utility Control Statements for DFSULTR0 ................. 243
Error Processing .......................................... 246
Examples of DFSULTR0 .................................. 247

Part 4. Analysis Utilities and Reports .......................... 253

Chapter 11. IMS Monitor Report Print Utility (DFSUTR20) .... 257
Restrictions for DFSUTR20 ............................... 257
Input and Output for DFSUTR20 ......................... 257
Chapter 12. File Select and Formatting Print Utility (DFSERA10) .......................... 261
Input and Output for DFSERA10 ................................................................. 261
JCL Requirements for DFSERA10 ............................................................... 262
Utility Control Statements for DFSERA10 ............................................... 263
Examples of DFSERA10 ............................................................................. 269
Record Format and Print Module (DFSERA30) ......................................... 275
Program Isolation Trace Record Format and Print Module (DFSERA40) .... 282
DL/I Call Image Capture Module (DFSERA50) ........................................... 286
IMS Trace Table Record Format and Print Module (DFSERA60) ............... 286
Enhanced Select Exit Routine (DFSERA70) ................................................ 287
Examples of Using the Enhanced Select Exit Routine (DFSERA70) .......... 289

Chapter 13. Fast Path Log Analysis Utility (DBFULTA0) .............................. 291
Restrictions for DBFULTA0 ........................................................................ 292
Input and Output for DBFULTA0 .............................................................. 292
Detail-Listing-of-Exception-Transactions Report ...................................... 294
Summary-of-Exception-Detail-by-Transaction-Code (for IFP Regions) Report 299
Overall-Summary-of-Transit-Times-by-Transaction-Code (for IFP-Regions) Report .......................................................... 300
Overall Summary of Resource Usage and Contentions for All Transaction Codes and PSBs Report ............................................................ 300
Summary-of-Region-Occupancy Report ..................................................... 302
Summary-of-VSO-Activity Report .............................................................. 303
Recapitulation-of-the-Analysis Report ....................................................... 304
JCL Requirements for DBFULTA0 ............................................................. 305
Utility Control Statements for DBFULTA0 .............................................. 306
Error Processing for DBFULTA0 .............................................................. 311

Chapter 14. Offline Dump Formatter Utility (DFSOFMD0) ............................ 313
Interactive Dump Formatter ....................................................................... 313
Migration Considerations .......................................................................... 314
Restrictions for DFSOFMD0 ...................................................................... 314
Environments for DFSOFMD0 ................................................................. 314
Input and Output for DFSOFMD0 ............................................................. 315
IPCS Execution ......................................................................................... 315

Chapter 15. Log Transaction Analysis Utility (DFSILTA0) ............................ 319
Restrictions for DFSILTA0 .......................................................................... 319
Input and Output for DFSILTA0 ............................................................... 320
JCL Requirements for DFSILTA0 ............................................................. 320

Chapter 16. Statistical Analysis Utility (DFSISTS0) ....................................... 325
Restrictions for DFSISTS0 .......................................................................... 325
Input and Output for DFSISTS0 ............................................................... 325
Examples of DFSISTS0 ............................................................................. 331
JCL Requirements for DFSISTS0 ............................................................. 335
Utility Control Statements for DFSISTS0 ................................................. 340

Part 5. Interpreting IMS Reports ................................................................. 343

Chapter 17. Interpreting IMS Monitor Reports ............................................. 347
Transaction Flow and IMS Monitor Events ............................................... 347
IMS Monitor Trace Event Intervals ........................................................... 350
## Figures

1. DBDGEN Input Record Structure (Except DEDB) ........................................................................ 10
2. DEDB DBDGEN Input Record Structure .................................................................................... 11
3. JCL for DBDGEN Utility ............................................................................................................. 12
4. Connections through Physical Child and Physical Twin Pointers .............................................. 31
5. Example of MVS Access Method Services Parameters from DBD Generation ....................... 87
6. Example of DBDGEN Input ........................................................................................................ 87
7. Segment Flag Codes .................................................................................................................. 88
8. Payroll and Skills Inventory Data Structures .......................................................................... 90
9. HSAM DBD Generation ............................................................................................................. 90
10. HISAM DBD Generations .......................................................................................................... 91
11. HDAM DBD Generation ........................................................................................................... 92
12. Summary of Statements for the Primary HIDAM Index Relationship ....................................... 94
13. HIDAM and Primary HIDAM Index DBD Generations .............................................................. 94
14. PHDAM DBD Generations ......................................................................................................... 96
15. PHIDAM DBD Generations ........................................................................................................ 96
16. GSAM DBD Generations .......................................................................................................... 96
17. Main Storage Database DBD Generations ............................................................................. 97
18. Data Entry Database DBD Generations .................................................................................. 99
19. DBD Generation of DEDB Subset Pointers Sample ................................................................. 100
20. Comparison of Unidirectional, Physically Paired Bidirectional, and Virtually Paired Bidirectional Logical Relationships ................................................................. 101
21. Logical Relationship Between Physical Databases and The Resulting Logical Databases That Can Be Defined ........................................................................................................ 103
22. DBD Generation Statements Examples .................................................................................. 104
23. Database Indexed by Two Secondary Indexes ........................................................................ 108
24. Indexed Database and Secondary Index Database ................................................................... 108
25. Database Indexed by Three Secondary Indexes in a Shared Secondary Index Database .......... 109
26. Indexed Database, Primary Index Database, and Shared Secondary Index Database DBD Generations .................................................................................................................................. 110
27. PSBGEN Procedure Statement ............................................................................................... 113
28. Procedure for Invoking PSBGEN ............................................................................................. 115
29. KEYLEN Definition .................................................................................................................. 126
30. Data Structure of Segment Definition .................................................................................... 131
31. Sample Hierarchic Data Structure ........................................................................................... 137
32. Sample Field Level Sensitivity PSB Generation ...................................................................... 140
33. A PSBGEN Statement Used to Define a DL/I Database Statement (Example 1) ................. 142
34. A PSBGEN Statement Used to Define a DL/I Database PCB Statement (Example 2) ........... 143
35. A PSBGEN PCB Statement Used to Define a DL/I Database PCB Statement (Example 3) .... 144
36. A PSBGEN Statement Used to Define a Logical Relationship and Produce Output ............ 144
37. The Data Structure and JCL For a Message Switching or Conversational Message Program .... 145
38. The Data Structure and JCL For a Logical Relationship in Database DI21PART .................... 146
39. The Data Structure and JCL For a Logical Database Defined From DL/I Database DI21PART .... 147
40. The Data Structure and JCL For a Logical Relationship in Database DI21PART That Produces Output (Part 1) ........................................................................................................ 148
41. The Data Structure and JCL for a Logical Relationship in Database DI21PART That Produces Output (Part 2) ........................................................................................................ 149
42. The Data Structure and JCL for a Logical Database Defined From DL/I Database DI21PART .... 150
43. Database Indexed by Three Secondary Indexes in a Shared Secondary Index Database .......... 151
44. The Data Structure and JCL For Index Through Segment DA ............................................... 151
45. The Data Structure and JCL For Index Through Segment DC ................................................ 152
46. The Data Structure and JCL For Index Through Segment DE ................................................ 153
47. Application Control Blocks Maintenance Utility ...................................................................... 157
48. ACBLIB Maintenance Procedure ............................................................................................. 157
104. Program-Summary Report .................................................. 361
105. Call-Summary Report ....................................................... 362
106. Program-I/O Report ........................................................... 364
107. Message Format Buffer Pool Report ....................................... 367
108. Message-Queue-Pool Report ................................................. 367
109. General Reports for SNAPQ Effects ........................................ 368
110. Transaction-Queuing Report ................................................ 369
111. Database-Buffer-Pool Report .............................................. 370
112. VSAM-Buffer-Pool Report .................................................. 370
113. Communication-Summary Report .......................................... 371
114. Line-Functions Report ....................................................... 371
115. Communication-Wait Report .............................................. 372
116. Pool-Space-Failure Report ................................................ 372
117. Deadlock-Event-Summary Report ......................................... 373
118. Latch-Conflict-Statistics Report .......................................... 375
119. Distribution-Appendix Report ............................................. 378
120. Number of Transactions Processed For Each Scheduling Of An Application Program ........................................... 379
121. MSC-Traffic Report .......................................................... 381
122. MSC-Summaries Report ..................................................... 382
123. MSC-Queuing-Summary Report .......................................... 383
124. IMS Monitor Trace Event Intervals ........................................ 387
125. IMS Monitor System Configuration Report and Trace Interval .......... 389
126. Run Profile Report ............................................................ 389
127. Region Summary Report ..................................................... 392
128. Region Wait Report ............................................................ 393
129. Programs-by-Region Report ................................................. 394
130. Event Intervals ................................................................. 395
131. Program Summary Report .................................................. 396
132. Call Summary Report .......................................................... 397
133. Transaction Queuing Report ............................................... 399
134. Database Buffer Pool Report ............................................... 400
135. VSAM Buffer Pool Report .................................................. 400
136. Pool Space Failure Report .................................................. 401
137. Deadlock Event Summary Report ......................................... 401
138. Latch Conflict Statistics Report .......................................... 402
139. Distribution Appendix Report ............................................. 405
140. IMS Monitor Trace Event Intervals ........................................ 408
141. IMS Monitor System Configuration Report and Trace Interval .......... 410
142. Run Profile Report ............................................................ 411
143. Region Summary Report ..................................................... 414
144. Region Wait Report ............................................................ 415
145. Programs-by-Region Report ................................................. 416
146. Elapsed Time Event Intervals .............................................. 417
147. Program Summary Report .................................................. 418
148. Call Summary Report .......................................................... 419
149. Program I/O Report ........................................................... 421
150. Message Format Buffer Pool Report ....................................... 424
151. Message Queue Pool Report ................................................. 424
152. General Reports for SNAPQ Effects ....................................... 425
153. Transaction Queuing Report ............................................... 426
154. Communication Summary Report ........................................ 427
155. Line-Functions Report ....................................................... 427
156. Communication Wait Report .............................................. 428
157. Pool Space Failure Report .................................................. 428
158. Latch Conflict Statistics Report .......................................... 429
159. Distribution Appendix Report ............................................. 432
160. Number of Transactions Processed For Each Scheduling Of An Application Program . . . . . . 433
161. MSC Traffic Report . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 435
162. MSC Summaries Report . . . . . . . . . . . . . . . . . . . . . . . . . . . 436
163. MSC Queuing Summary Report . . . . . . . . . . . . . . . . . . . . . . . . . 437
164. PST-Accounting Report . . . . . . . . . . . . . . . . . . . . . . . . . . . . 440
165. VSAM-Buffer-Pool Report . . . . . . . . . . . . . . . . . . . . . . . . . . . 441
166. OSAM-Buffer-Pool Report . . . . . . . . . . . . . . . . . . . . . . . . . . . 443
167. Sequential-Buffering-Summary Report . . . . . . . . . . . . . . . . . . . . . . 445
168. Sequential-Buffering-Detail Report Page A . . . . . . . . . . . . . . . . . . . . . 446
169. Sequential-Buffering-Detail Report Page B . . . . . . . . . . . . . . . . . . . . . 448
170. Sequential-Buffering-Detail Report Page C . . . . . . . . . . . . . . . . . . . . . 451
171. Line-and-Terminal Report . . . . . . . . . . . . . . . . . . . . . . . . . . . 457
172. Transaction Report . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 457
173. Transaction Response-Report . . . . . . . . . . . . . . . . . . . . . . . . . . 458
174. Messages—Program-to-Program Reports . . . . . . . . . . . . . . . . . . . . . . 458
175. Messages—Queued-But-Not-Sent Reports . . . . . . . . . . . . . . . . . . . . . . 459
176. Messages Report . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 460
177. Log-Analysis Report . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 461
178. Application-Accounting Report . . . . . . . . . . . . . . . . . . . . . . . . . 464
### Tables

1. How to Read Syntax Diagrams .............................................................. xvi
2. DBD Generation Statement Instruction Summary .................................. 13
3. Using the Label Field to Group Segment Types ................................... 32
4. BLOCK= and RECORD= Operands ....................................................... 37
5. Use of POINTER= Operand Parameters (No Logical Relationship) .......... 61
6. Sample Concatenated Key for an Index Source Segment Type .................. 79
7. Same Index Source and Target Segment Types ..................................... 105
8. Different Index Source and Target Segment Types ............................... 106
9. Shared Secondary Index Database DBD Generation ................................ 107
10. How A KEYLEN Is Determined .......................................................... 125
11. Using LANG= Option in an LE/370 Environment for PL/I Compatibility ... 133
12. Allocation Information Priorities ....................................................... 167
13. Matrix Secured Resources for Variable I .......................................... 185
14. Matrix Secured Resources for Variable R .......................................... 185
15. Security Maintenance Utility Input Statements .................................... 189
17. Lock Name In A FP Database ............................................................ 276
18. PI Lock Compatability Matrix ........................................................... 279
19. IRLM Resultant State Matrix ............................................................ 279
20. IRLM Compatibility Matrix ............................................................... 280
21. Transaction Flow and IMS Monitor Events Description ....................... 347
22. IMS Monitor Reports Output Sequence and Information ....................... 351
23. Distribution Reports by Region Summary .......................................... 375
24. Report Distributions by Program Region .......................................... 376
25. Report Distributions by Program Summary ........................................ 376
26. Report Distributions by Communication Summary .............................. 376
27. Report Distributions by Line Functions ............................................. 376
28. Report Distributions by MSC Queuing Summary ................................ 376
29. Report Distributions by Transaction Queuing ..................................... 377
30. Report Distributions by Call Summary ............................................. 377
31. Wait Time Distributions ................................................................. 377
32. IMS Monitor Reports Output Sequence and Information ....................... 387
33. Report Distributions by Region Summary .......................................... 403
34. Report Distributions by Programs Region ......................................... 403
35. Report Distributions by Program Summary ........................................ 403
36. Report Distributions by Call Summary ............................................. 403
37. Wait Time Distributions ................................................................. 404
38. Output Sequence and Information from IMS Monitor Reports ............... 408
39. Report Distributions by Region Summary .......................................... 430
40. Report Distributions by Program Region .......................................... 430
41. Report Distributions by Program Summary ........................................ 430
42. Report Distributions by Communication Summary .............................. 430
43. Report Distributions by Line Functions ............................................. 430
44. Report Distributions by MSC Queuing Summary ................................ 431
45. Report Distributions by Transaction Queuing ..................................... 431
46. Report Distributions by Call Summary Queueing ............................... 431
47. Wait Time Distributions ................................................................. 431
48. Log-Analysis Report Line Format ..................................................... 461

© Copyright IBM Corp. 1974, 2005
About This Book

This book is a reference manual for database administrators and system programmers who use the IMS™ utilities common to the IMS Database Manager (IMS DB) and IMS Transaction Manager (IMS TM) to administer the IMS system. This book can also be used by people interested in using these utilities with CICS/MVS® or CICS/ESA®.

This book is one of two utilities references in the IMS library. The scope of the publications is as follows:

- **IMS Version 8: Utilities Reference: System** describes utilities that apply to IMS at a system level or that affect both database and data communications operations.
- **IMS Version 8: Utilities Reference: Database and Transaction Manager** describes utilities that affect database operations and data communications.

This softcopy book is available in PDF and BookManager® formats, and also as part of the DB2® Information Management Software Information center for z/OS® Solutions. To view the information within the DB2 Information Management Software Information Center for z/OS Solutions, go to publib.boulder.ibm.com/infocenter/db2zhelp. To get the most current versions of the PDF and BookManager formats, go to the IMS Library page at www.ibm/software/data/ims/library.html.

Organization of This Book

This book has six parts:

- **Part 1, “Generation Utilities”** contains information on the generation utilities for DBDs, PSBs, and ACBs.
- **Part 2, “Service Utilities”** describes the service utilities for dynamic allocation, security maintenance, and online change.
- **Part 3, “Log Utilities”** has information on the utilities used for archiving, merging, and recovering logs.
- **Part 4, “Analysis Utilities and Reports”** discusses the utilities used to generate and print IMS reports.
- **Part 5, “Interpreting IMS Reports”** explains how to interpret IMS reports.
- **Part 6. Appendixes** contains the Bibliography and the Index.

For a complete list of all books cited in this manual see the “Bibliography” on page 471.

Prerequisite Knowledge

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

The reader should be familiar with OS/390®, and with IMS concepts, facilities, and access methods. The prerequisite publications are:

- **IMS Version 8: Release Planning Guide**
- **IMS Version 8: Administration Guide: System**
- **IMS Version 8: Administration Guide: Database Manager**
Organization of Utility Descriptions

Utility descriptions are generally organized the same way, to help you find information easily. Most utilities are described this way:

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

When applicable, the descriptions also include:

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

How to Read Syntax Diagrams

This book contains syntax diagrams.

Each syntax diagram begins with a double right arrow and ends with a right and left arrow pair. Lines that begin with a single right arrow are continuation lines. You read a syntax diagram from left to right and from top to bottom, following the direction of the arrows.

Table 1 describes the conventions that are used in syntax diagrams in this information:

<table>
<thead>
<tr>
<th>Table 1. How to Read Syntax Diagrams</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convention</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td><img src="image" alt="Diagram 1" /></td>
</tr>
<tr>
<td><img src="image" alt="Diagram 2" /></td>
</tr>
<tr>
<td><img src="image" alt="Diagram 3" /></td>
</tr>
<tr>
<td><img src="image" alt="Diagram 4" /></td>
</tr>
</tbody>
</table>
### Table 1. How to Read Syntax Diagrams (continued)

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Diagram" /></td>
<td>You have the option to specify A, B, C, or none of these values. If you don’t specify a value, A is the default.</td>
</tr>
<tr>
<td><img src="image2" alt="Diagram" /></td>
<td>You have the option to specify one, more than one, or none of the values A, B, or C. Any required separator for multiple or repeated values (in this example, the comma) is shown on the arrow.</td>
</tr>
<tr>
<td><img src="image3" alt="Diagram" /></td>
<td>You have the option to specify value A multiple times. The separator in this example is optional.</td>
</tr>
<tr>
<td><img src="image4" alt="Diagram" /></td>
<td>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.</td>
</tr>
</tbody>
</table>

- **Punctuation marks and numbers**: Enter punctuation marks (slashes, commas, periods, parentheses, quotation marks, equal signs) and numbers exactly as shown.
- **Uppercase values**: Keywords, their allowable synonyms, and reserved parameters appear in uppercase letters for OS/390. Enter these values exactly as shown.
- **Lowercase values**: Keywords, their allowable synonyms, and reserved parameters appear in lowercase letters for UNIX®. Enter these values exactly as shown.
- **Lowercase values in italic (for example, name)**: Supply your own text or value in place of the name variable.

A b symbol indicates one blank position.

**Other syntax conventions include the following:**

- When you enter commands, separate parameters and keywords by at least one blank if there is no intervening punctuation.
- Footnotes are shown by a number in parentheses, for example, (1).
- Parameters with number values end with the symbol #.
- Parameters that are names end with ‘name’.
- Parameters that can be generic end with the symbol *.
Example Syntax Diagram

Here is an example syntax diagram that describes the hello command.

```
// hello
//   Name
//   Greeting
```

**Name:**

```
(1)
```

**Greeting:**

```
(2)
```

**Notes:**

1. You can code up to three names.
2. Compose and add your own greeting (for example, how are you?).

According to the syntax diagram, these commands are all valid versions of the hello command:

```
hello
hello name
hello name, name
hello name, name, name
hello, your_greeting
hello name, your_greeting
hello name, name, your_greeting
hello name, name, name, your_greeting
```

The space before the name value is significant. If you do not code name, you must still code the comma before your_greeting.

CICS, DBCTL, and DCCTL

When running CICS® with DBCTL, CICS/ESA 3.1 or later releases must be used.

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

For DCCTL users, all utilities, commands, and parameters that are valid for IMS/TM are valid for DCCTL, unless otherwise noted.

How to Send Your Comments

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

- Go to the IMS home page at www.ibm.com/ims. There you will find an online feedback page where you can enter and submit comments.
• Send your comments by e-mail to imspubs@us.ibm.com. Be sure to include the title, the part number of the title, the version of IMS, and, if applicable, the specific location of the text you are commenting on (for example, a page number in the PDF or a heading in the Information Center).
Summary of Changes

Changes to the Current Edition of This Book for IMS Version 8
This edition, which is available in softcopy format only, includes technical and editorial changes.

Changes to This Book for IMS Version 8
This book contains new and changed information about the following enhancements:
- Global Online Change utility (DFSUOLC0)
- Online Change Copy utility (DFSUOCU0)
- Increase in available Areas in data entry databases (DEDBs) from 240 to 2048

Library Changes for IMS Version 8
Changes to the IMS Library for Version 8 include the addition of new titles, the elimination of one title, organizational changes, and accessibility enhancements. Changes are indicated by a vertical bar (|) to the left of the changed text.

New, Revised, and Eliminated Titles
The following list details major changes to the IMS Version 8 library:
- **IMS Version 8: Common Service Layer Guide and Reference**
  The library includes new information: **IMS Version 8: Common Service Layer Guide and Reference** (CSL). This information is available only in PDF and BookManager formats.
  - **IMS Version 8: Base Primitive Environment Guide and Reference**
  - **IMS Version 8: Common Queue Server Guide and Reference**
- The information formerly titled **IMS Version 7: Common Queue Server and Base Primitive Environment Guide and Reference** has been divided in the IMS Version 8 library:
  - **IMS Version 8: Base Primitive Environment Guide and Reference**
  - **IMS Version 8: Common Queue Server Guide and Reference**
- The information formerly titled **IMS Version 7: Installation Volume 1: Installation and Verification** is now titled **IMS Version 8: Installation Volume 1: Installation and Verification**. All installation information is now in the IMS Version 8 Program Directory.
- **IMS Version 8: Sample Operating Procedures**
  This information is no longer produced for the IMS library from IMS Version 8 and after.
  - The information formerly titled **IMS Version 8: IMS Java™ User’s Guide** is now titled **IMS Version 8: IMS Java Guide and Reference**

Organizational Changes
Organizational changes to the IMS Version 8 library include changes to:
- **IMS Version 8: Application Programming: Database Manager**
- **IMS Version 8: Application Programming: EXEC DLI Commands for CICS and IMS**
- **IMS Version 8: Application Programming: Transaction Manager**
- **IMS Version 8: Messages and Codes, Volume 1**

The section titled “DL/I Status Codes” will now only appear in IMS Version 8: Messages and Codes, Volume 1.

The section titled “MFS Language Utility” has been renamed to “MFS Language Utility Control Statements” and has been moved from IMS Version 8: Application Programming: Transaction Manager to IMS Version 8: Utilities Reference: Database and Transaction Manager.

Deleted Information
OS/390 does not support the Virtual Fetch function any longer. Consequently, all information associated with Virtual Fetch has been deleted from the following IMS Version 8 information:

- IMS Version 8: Administration Guide: System
- IMS Version 8: Failure Analysis Structure Tables (FAST) for Dump Analysis
- IMS Version 8: Installation Volume 2: System Definition and Tailoring
- IMS Version 8: Messages and Codes, Volume 1
- IMS Version 8: Messages and Codes, Volume 2

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

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

User Assistive Technologies
Assistive technology products, such as screen readers, function with the user interfaces found in IMS. Consult the assistive technology documentation for specific information when using it to access these interfaces.

Accessible Information
Online information for IMS Version 8 is available in BookManager format, which is an accessible format. All BookManager functions can be accessed by using a keyboard or keyboard shortcut keys. BookManager also allows you to use screen readers and other assistive technologies. The BookManager READ/MVS product is included with the OS/390 base product, and the BookManager Library Reader™ (for workstations) is available on the IMS Licensed Product Kit (CD), which is available for downloading from IBM at www.ibm.com

Keyboard Navigation of the User Interface
Users can access IMS user interfaces using TSO/E or ISPF. Refer to the z/OS: TSO/E Primer, z/OS: TSO/E User’s Guide, z/OS: ISPF User’s Guide. These guides describe how to navigate each interface, including the use of keyboard...
shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.
Part 1. Generation Utilities

Chapter 1. Database Description (DBD) Generation

Information Specified in DBD Generation

DBD Generation for Database Types

HSAM DBD Generation

GSAM DBD Generation

HISAM DBD Generation

HDAM and PHDAM DBD Generation

HIDAM and PHIDAM DBD Generation

MSDB DBD Generation

DEDB DBD Generation

Index and PSINDEX DBD Generation

Logical DBD Generation

DBD Generation Input Record Structure (Except for DEDB DBDs)

DEDB DBD Generation Input Record Structure

DBD Generation Coding Conventions

DBDGEN Procedure

PROC Statement

JCL Parameters

DBDGEN Statements

DBD Statement

DBD Statement Parameter Descriptions

DATASET Statements

DATASET Statement Parameter Description

Data Sets in IMS Data Set Groups

AREA Statement

AREA Statement Parameter Description

SEGM Statement

SEGM Statement for PSINDEX Databases

LCHILD Statement

LCHILD Statement Parameter Description

FIELD Statement

FIELD Statement Parameter Description

XDFLD Statement

XDFLD Statement Parameter Description

DBDGEN, FINISH, and END Statements

Output for DBD Generation

Control Statement Listing

DBD Generation Error Conditions

Examples of DBD Generation

Examples without Secondary Index or Logical Relationships

Summary of Physical Database Description Examples

Examples with Logical Relationships

Examples with Secondary Indexes

Chapter 2. Program Specification Block (PSB) Generation

Input and Output of PSB Generation

PSBGEN Procedure

PROC Statement

Step C

Step L

Invoking the Procedure

Utility Control Statements for PSB Generation

Alternate PCB Statement
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DL/I or Fast Path Database PCB Statement</td>
<td>117</td>
</tr>
<tr>
<td>GSAM PCB Statement</td>
<td>128</td>
</tr>
<tr>
<td>SENSEG Statement</td>
<td>129</td>
</tr>
<tr>
<td>SENFLD Statement</td>
<td>131</td>
</tr>
<tr>
<td>PSBGEN Statement</td>
<td>132</td>
</tr>
<tr>
<td>END Statement</td>
<td>136</td>
</tr>
<tr>
<td>Output Messages and Statistics for PSB Generation</td>
<td>136</td>
</tr>
<tr>
<td>Examples of PSB Generation</td>
<td>137</td>
</tr>
<tr>
<td>PSB Generation Examples</td>
<td>137</td>
</tr>
<tr>
<td>Field Level Sensitivity PSB Generation Example</td>
<td>139</td>
</tr>
<tr>
<td>Fast Path PSB Generation Examples</td>
<td>140</td>
</tr>
<tr>
<td>Additional PSB Generation Examples</td>
<td>141</td>
</tr>
<tr>
<td>Examples of a Sample Problem with an Application Database</td>
<td>144</td>
</tr>
<tr>
<td>Example of a Shared Secondary Index</td>
<td>150</td>
</tr>
<tr>
<td><strong>Chapter 3. Application Control Blocks Maintenance Utility</strong></td>
<td>155</td>
</tr>
<tr>
<td>Restrictions for ACBGEN</td>
<td>156</td>
</tr>
<tr>
<td>Input and Output for ACBGEN</td>
<td>156</td>
</tr>
<tr>
<td>ACBGEN Procedure</td>
<td>157</td>
</tr>
<tr>
<td>EXEC Statement</td>
<td>158</td>
</tr>
<tr>
<td>DD Statements</td>
<td>158</td>
</tr>
<tr>
<td>DFSACBCP Control Statement</td>
<td>159</td>
</tr>
<tr>
<td>Utility Control Statements for ACBGEN</td>
<td>159</td>
</tr>
<tr>
<td>Managing Dynamic Option (DOPT) PSBs</td>
<td>162</td>
</tr>
<tr>
<td>Error Processing for ACBGEN</td>
<td>163</td>
</tr>
<tr>
<td>Examples of ACBGEN</td>
<td>163</td>
</tr>
<tr>
<td>Example 1</td>
<td>163</td>
</tr>
<tr>
<td>Example 2</td>
<td>163</td>
</tr>
<tr>
<td>Example 3</td>
<td>163</td>
</tr>
</tbody>
</table>
Chapter 1. Database Description (DBD) Generation

Use the Database Description Generation (DBDGEN) utility to define a database so it can be used by an application program. You create a Database Description (DBD) by coding special macro instructions. These macros become the input to the DBDGEN utility. Use DBDGEN for the following types of databases:

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

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

The DBDGEN program accepts several types of control statements. Each control statement type is briefly described as follows:

- 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 control 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 control statement.
- The SEGM statement defines the specified database’s segments. The SEGM statement is used with the following statements:
  - FIELD
  - XDFLD
  - LCHILD
  Each statement defines different aspects of 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 MVS™ assembler that the end of the input statements has been reached.

Related Reading: For more information on High Availability Large Databases, see IMS Version 8: Administration Guide: Database Manager.

In this Chapter:

- “Information Specified in DBD Generation” on page 4
- “DBD Generation for Database Types” on page 4

© Copyright IBM Corp. 1974, 2005
A database description (DBD) is a DL/I control block containing all of the database information needed by an application program. You can use only one physical DBD to describe each physical database; otherwise, user abend U850 or U853 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

You can also use the DBDGEN utility to define the name and data options of selected exit routines.

**DBD Generation for Database Types**

The following databases use DBDGEN:

- HSAM (including SHSAM)
- GSAM
- HISAM (including SHISAM)
- HDAM
- PHDAM
- HIDAM
- PHIDAM

SHSAM and SHISAM are simple databases. Each contains only one fixed-length segment type. Discussions on SHSAM and SHISAM can be found in paragraphs dealing with HSAM and HISAM, respectively.

The following also use DBDGEN:

- MSDB
- DEDB
- Index
  - Primary HIDAM
  - Secondary
- PSINDEX
- Logical

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

For an HSAM database you cannot specify:
• The use of hierarchic or physical child/physical twin pointers between segments in the database
• The use of logical or index relationships between segments

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

You cannot specify:
• SEGM and FIELD statements
• The use of logical or index relationships between segments

IMS™ adds 2 bytes to the record length value specified in the DBD in order to accommodate the ZZ field that is needed to make up the BSAM RDW. Whenever the database is GSAM/BSAM and the records are variable (V or VB), IMS adds 2 bytes. The record size of the GSAM database is 2 bytes greater than the longest segment that is passed to IMS by the application program.

HISAM DBD Generation

During DBD generation for a HISAM 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. When defined, 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.
DBD Generation for Database Types

- 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™ end 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/physical twin pointers between segments in a HISAM database.

### HDAM and 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 (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
- The use of hierarchic or physical child/physical twin pointers between segments in the database
- Logical relationships (optional) between segments using direct address and/or symbolic pointer options
- From 0 to 255 fields for each segment type, and a maximum of 1000 for the 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 end users access to updated IMS data

**Restrictions of 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. For more information on defining partitions, see the [IMS Version 8: Administration Guide: Database Manager](#).

### 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
DBD Generation for Database Types

- 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/physical twin pointers between segments in the database
- Logical relationships (optional) between segments using direct address and/or symbolic pointer options
- From 0 to 255 fields for each segment type, and a maximum of 1000 for the 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 end users access to updated IMS data

DBDGEN for PHDAM:
- 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. For more information on defining partitions, see the IMS Version 8: Administration Guide: Database Manager

MSDB DBD Generation

During DBD generation for a 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

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 MSDBCPCp data set. In this case, the headers in the MSDBCPCp 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 a 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 end 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
- Fields used with secondary indexes

**Index and PSINDEX 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 prime 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.

**Restrictions:**
- You cannot specify any additional FIELD statements as you might for a secondary index.
- You cannot use DBDGEN to define individual partitions. For more information on defining partitions, see [ IMS Version 8: Administration Guide: Database Manager ]
- Nonunique 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, HDAM, PHDAM, HIDAM, or PHIDAM databases.

During DBD generation for a 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.
• From 1 to 16 segment types.
• From 1 to 16 secondary index relationships.
• From 1 to 1000 fields for each segment type.

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 ten types of control statements. Each control statement must be added to the SYSIN input stream in a specific order. Figure 1 on page 10 shows the rules for structuring DBD generation input.

**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 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, and XDFLD statements 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.

Chapter 1. Database Description (DBD) Generation 9
DBD Generation Input Record Structure

**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.

![Diagram of DBD Generation Input Record Structure](image)

**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 not permitted.
- Logical relationships between databases are not permitted.
- LCHILD and XDFLD statements are not permitted.
- Sequential dependent segments cannot have dependents.
A separate input set of records is required for each database.

Figure 2 shows the rules for structuring a DEEDB DBD generation input set of records.

**DBD Generation Coding Conventions**

DBD generation statements are assembler language macro instructions and therefore are subject to the rules contained in the *High Level Assembler Programmer’s Guide*.

Each control statement must be identified by an operation code, for example: record-type code.

**DBDGEN Procedure**

Stage 2 of system definition causes the DBDGEN procedure to be placed in the IMS.PROCLIB library.

This is a two step assemble and link-edit procedure to produce database definition blocks (DBDs).

**PROC Statement**

An example of the JCL for the DBDGEN utility is shown in Figure 3 on page 12.
**DBDGEN Procedure**

```
//  PROC MBR=TEMPNAME,SOUT=A,RGN=4M,SYS2=
//  EXEC PGM=ASMA90,REGION=&RGN,PARM='OBJECT,NODECK'
//SYSLIB DD DSN=IMS.&SYS2SDFSMAC,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)
//SYST1 DD UNIT=SYSDA,DISP=(,DELETE),
//  SPACE=(CYL,(10,5))
//L EXEC PGM=IEWL,PARM='XREF,LIST',COND=(0,LT,C),REGION=120K
//SYSLIN DD DSN=*.*,DISP=(OLD,DELETE)
//SYSPRINT DD SYSOUT=&SOUT,DCB=BLKSIZE=1089,
//  SPACE=(121,(90,90),RLSE)
//SYSLMOD DD DSN=IMS.&SYS2BDLIB(MBR),DISP=SHR
//SYST1 DD UNIT=(SYSDA,SEP=(SYSLMOD,SYSLIN)),
//  SPACE=(1024,(100,10),RLSE),DISP=(,DELETE)
```

Figure 3. JCL for DBDGEN Utility

**Invoking the Procedure**

To process a request for a DBDGEN, the DBD generation control statements must be created and appended to the JCL (shown in Figure 4 on page 31 which invokes the DBDGEN procedure.

```
//DBDGEN JOB MSGLEVEL=1
// 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 256KB.

**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
operand must be enclosed in quotes and must include a trailing period; for
example, SYS2=’IMS.’.

**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.

**Related Reading:** Refer to *High Level Assembler Programmer’s Guide* for
information on assembling steps.

**Step L**
Step L is the link-edit 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 link-edit 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 link-edit step with RMODE=24.

**Related Reading:** Refer to *MVS/DFP Linkage Editor and Loader* for more
information about linkage editors.

**DD Statements:**

IMS.DBDLIB DD
Defines an output partitioned data set, IMS.DBDLIB, for the linkage editor.

---

**DBDGEN Statements**

Table 2 shows the statement instruction types used as input to the DBD generation
utility to define a database. Also included is the general use of each statement and
the number of each type used per DBD generation.

<table>
<thead>
<tr>
<th>Statement</th>
<th>General Use</th>
<th>Number used per DBD generation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>HSAM</td>
</tr>
<tr>
<td>[PRINT]</td>
<td>Controls printing of assembly listing if present</td>
<td>0-1</td>
</tr>
<tr>
<td>DBD</td>
<td>Defines database name</td>
<td>1</td>
</tr>
<tr>
<td>DATASET</td>
<td>Defines a data set group within a database</td>
<td>1</td>
</tr>
</tbody>
</table>
### DBDGEN Statements

Table 2. DBD Generation Statement Instruction Summary (continued)

<table>
<thead>
<tr>
<th>Statement</th>
<th>General Use</th>
<th>Number used per DBD generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AREA</td>
<td>Defines an area within a Fast Path database</td>
<td>HSAM</td>
</tr>
<tr>
<td>SEGM</td>
<td>Defines a segment type within a data set group or area</td>
<td>1-255</td>
</tr>
<tr>
<td>[LCHILD]</td>
<td>Defines a logical or index relation—ship between segment types</td>
<td>0</td>
</tr>
<tr>
<td>[FIELD]</td>
<td>Defines a field within a segment type</td>
<td>0-1000</td>
</tr>
<tr>
<td>[XDFLD]</td>
<td>Defines fields used with secondary indexes</td>
<td>0</td>
</tr>
</tbody>
</table>

| DBDGEN     | Indicates the end of DBD generation statements | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| FINISH     | Checks for successful DBD generation | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| END        | Indicates end of DBD generation input to the MVS assembler | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Notes:**
1. For operand information, see *OS/VS-DOS/VSE-VM/370 Assembler Language*
3. The maximum combined total of FIELD and XDFLD statements per DBD generation is 1000.
5. All Full Function Database names and DEDB area names must be unique.

### DBD Statement

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. A description of the statement parameters is in "DBD Statement Parameter Descriptions" on page 22.

For details on the coding format for Assembler macro instructions, refer to the "Assembler Coding Conventions" section in the *IBM Assembler Manual*, publication number SC26-4940-03.
DBD Statement for HSAM Database

```
DBD NAME=dbname1,ACCESS=HSAM,SHSAM,DATXEXIT=YES
```

DBD Statement for GSAM Database

```
DBD NAME=(dbname1),ACCESS=GSAM,BSAM,PASSWD=YES,DATXEXIT=YES
```

DBD Statement for HISAM Database

```
DBD NAME=(dbname1),ACCESS=HISAM,SHISAM,PASSWD=YES,EXIT=(A C),VERSION='n',DATXEXIT=YES
```

A:

```
exitname NOLOG LOG NOLOG B NOLOG,LOG
```

B:

```
KEY NOKEY PATH NOPATH DATA NODATA
```
DBDGEN Statements

C:

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

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 is not required because only logging is being requested, specify the exit name as and the default logging parameter is LOG. If you do specify an exit routine name, the default logging parameter is NOLOG.
4. Used to control the CASCADE options.

DBD Statement for HDAM Database

A: 

EXIT=(4)

B: 

exitname=NOLOG

PASSWD=NO
DATXEXIT=YES

NAME=(dbname1), ACCESS=(HDAM, OSAM)
RMNANE=(mod, anch, rbn, bytes)
EXIT=NONE
VERSION='n'

Utilities Reference: System
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 is not required because only logging is being requested, specify the exit name as * and the default logging parameter is LOG. If you do specify an exit routine name, the default logging parameter is NOLOG.

5 Used to control the CASCADE options.

DBD Statement for PHDAM Database

A:

C:
**DBDGEN Statements**

```plaintext
exitname
```

**B:**

```plaintext
,KEY,PATH,DATA
```

**C:**

```plaintext
,CASCADE
```

### 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 is not required because only logging is being requested, specify the exit name as * and the default logging parameter is LOG. If you specify an exit routine name, the default logging parameter is NOLOG.
5. Used to control the CASCADE options.

**DBD Statement for HIDAM Database**

```plaintext
DBD--NAME=dbname1,ACCESS=(HIDAM,OSAM),VSAM
```

```
EXIT=(NONE)
```

```
,VERSION='n'
```

---

18 Utilities Reference: System
**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 is not required because only logging is being requested, specify the exit name as * and the default logging parameter is LOG. If you do specify an exit routine name, the default logging parameter is NOLOG.

4. Used to control the CASCADE options.

**DBD Statement for PHIDAM Database**
**DBDGEN Statements**

**A:**

- **exitname**
  - **NOLOG**
  - **LOG**
  - **NOLOG**

**B:**

- **KEY**
- **NOKEY**
- **PATH**
- **NOPATH**
- **DATA**
- **NODATA**

**C:**

- **CASCADE**
- **NOCASCADE**
- **(CASCADE B)**
- **(NOCASCADE B)**

**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 is not required because only logging is being requested, specify the exit name as * and the default logging parameter is LOG. If you specify an exit routine name, the default logging parameter is NOLOG.

4. Used to control the CASCADE options.

**DBD Statement for MSDB Database**

```
DBD--NAME=dbname1, ACCESS=MSDB
```
DBD Statement for DEDB Database

```
| DBD — NAME = dbname1 — , ACCESS = DEDB — , RMNAME = — (mod, ... XCI) |
```

A:

```
| EXIT = — (1) NONE — ( — A — | C — ) — (2) , VERSION = ’n’ |
```

B:

```
| , KEY — , NOKEY — , PATH — , NOPATH — , DATA — , NODATA — |
```

C:

```
| , CASCADE — , NOCASCADE — , (CASCADE — B — ) — , (NOCASCADE — B — ) |
```

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 is not required because only logging is being requested, specify the exit name as * and the default logging parameter is LOG. If you do specify an exit routine name, the default logging parameter is NOLOG.
4. Used to control the CASCADE options.

DBD Statement for INDEX Database

```
| DBD — NAME = (dbname1 — , dbname2 — ) |
```

Chapter 1. Database Description (DBD) Generation
DBDGEN Statements

```
 DBDGEN Statements

 Notes:
 1  A secondary index must use VSAM.
```

```
 DBD Statement for PSINDEX Database

 Notes:
 1  A secondary index must use VSAM.
```

```
 DBD Statement for LOGICAL Database

 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 DBD for the database being described. The name can be from 1 to 8 alphanumeric characters and can be the same as that specified in the DD1= operand of the first DATASET control statement. For a shared secondary index database, the names of up to 16 secondary index DBDs can be specified.

 Do not give a DBD the same name as an existing PSB. Using an existing name can cause unpredictable results. An error occurs at ACBGEN time.

 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 value of the operand has the following meaning:

 HSAM
 Means the hierarchical sequential access method (HSAM) is to be used for
the database described by this DBD. When HSAM is specified, and only one segment type is defined in the HSAM database, this operand defaults to SHSAM.

**SHSAM**
Specifies a 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**
Means the generalized sequential access method (GSAM) is to be used for the database described by the DBD. BSAM or VSAM can be specified as the operating system access method. VSAM is the default. When GSAM is specified, no SEGMENT control statement is allowed in the DBD generation.

**HISAM**
Means the hierarchical index sequential access method (HISAM) is to be used for the database described by this DBD. VSAM can be specified as the operating system access method. It is the default.

**SHISAM**
Specifies a 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**
Means the hierarchical direct access method (HDAM) is to be used for the database described by this DBD. OSAM or VSAM can be specified as the operating system access method. VSAM is the default.

**PHDAM**
Means the partitioned hierarchical direct access method (PHDAM) database is to be used for the database described by the DBD. OSAM or VSAM can be specified as the operating system access method. VSAM is the default.

**HIDAM**
Means the hierarchical indexed direct access method (HIDAM) is to be used for the database described by the DBD. OSAM or VSAM can be specified as the operating system access method. VSAM is the default.

**PHIDAM**
Means the partitioned hierarchical indexed direct access method (PHIDAM) database is to be used for the database described by the DBD. OSAM or VSAM can be specified as the operating system access method. VSAM is the default.

**MSDB**
Means a main storage database (MSDB) is described by the DBD.

**DEDB**
Means a data entry database (DEDB) is described by the DBD.

**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 operand is not used to define the primary index of a PHIDAM database.

**PROT or NOPROT**

Applies only to secondary index databases. The PROT operand on the DBD statement is an optional parameter that is used to ensure the integrity of all fields in index pointer segments that are used by IMS. Use of this parameter prevents an application program from doing a replace operation on any field within an index pointer segment except for fields within the user data portion of index pointer segments. When PROT is specified, delete operations are still enabled for index pointer segments. If PROT is specified and a delete is issued for an index pointer segment, the index target segment pointer in the index pointer segment is deleted. However, the index source segment that caused the index pointer segment to be created originally is not deleted. If NOPROT is specified, an application program 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. PROT is the default for this parameter.

**DOSCOMP**

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. Selection of the DOSCOMP operand 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. The DOSCOMP operand can only be specified for databases that use VSAM. The DOSCOMP operand is not supported for PHDAM, PHIDAM, or PSINDEX databases.

**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.

**PROT or NOPROT**

Applies only to secondary index databases. The PROT operand on the DBD statement is an optional parameter that is used to ensure the integrity of all fields in index pointer segments that are used by IMS. Use of this parameter prevents an application program from doing a replace operation on any field within an index pointer segment except for fields within the user data portion of index pointer segments. When PROT is specified, delete operations are still enabled for index pointer segments. If PROT is specified and a delete is issued for an index pointer segment, the index target segment pointer in the index pointer segment is deleted. However, the index source segment that caused the index pointer segment to be created originally is not deleted. If NOPROT is specified, an application program 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. PROT is the default for this parameter.

**LOGICAL**

Means that the database described by this DBD is a LOGICAL database. A LOGICAL database is composed of one or more physical databases. A LOGICAL DBD generation is meaningful only when physical DBD generations exist that define the segment types that are referenced by SEGM statements in a LOGICAL DBD generation.
PSNAME=
Specifies the module that selects the HALDB partition for PSINDEX, PHDAM, or PHIDAM databases. The operand parameter is a HALDB partition selection exit routine module name. This operand is only valid when ACCESS=PSINDEX, PHDAM, or PHIDAM is specified.

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

RMNAME=
Specifies information used to manage data stored in a DEDB or in the primary data set group of an HDAM or PHDAM database. This operand is only valid when ACCESS=HDAM, PHDAM, or DEDB is specified. The parameters of this operand are defined in the list that follows. 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 DBD generation-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
Specifies 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 mod parameter and 2 in the anchor point parameter.

Related Reading: Refer to IMS Version 8: Customization Guide for further examples of HDAM, PHDAM, and Fast Path DEDB randomizing modules.

anch
Specifies the number of root anchor points desired in each control interval or block in the root addressable area of an HDAM or PHDAM database. The default value of this parameter is one. The anch operand must be an unsigned decimal integer and must not exceed 255. Typical values are from 1 to 5. Select a two-stage randomizer by specifying the randomizer name in the mod parameter and 2 in the anchor point parameter.

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

The number of root anchor point for the DEDB is always 1.
**DBD PCB Macro Parameter Descriptions**

*rbn*

Specifies the maximum relative block number value that the user wants 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. The rbn operand must be an unsigned decimal integer whose value does not exceed $2^{24}-1$. If this parameter is omitted, no upper limit check is performed on the rbn created by the randomizing module. If this parameter is specified, but the user’s randomizing module produces an rbn greater than this parameter, the highest control interval or block in the root addressable area is used by IMS. If a user randomizing module produces a block number of zero, control interval or block one is used by IMS.

In an HDAM, PHDAM, HIDAM, or PHIDAM OSAM data set, the first bit map is in the first block of the first extent of the data set. In an HDAM, PHDAM, HIDAM, or PHIDAM 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 operand must be an unsigned decimal integer whose value does not exceed $2^{24}-1$. When the “rbn” parameter is omitted, the “bytes” parameter is ignored, which in turn, leaves no limit on the number of bytes of a database record that can be inserted into the root addressable area.

If the “bytes” operand 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 requires that enough space be available after the current EOF to contain the remainder of all database records that exceed the “bytes” specification. 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 that 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**

Causes DL/I open to use the DBDNAME for this DBD as the VSAM password when opening any data set for this database. This parameter is
only valid for DBDs that use VSAM as the access method. PASSWD=YES is invalid for ACCESS=LOGICAL, MSDB, or DEDB. When the user defines the VSAM data sets for this database using the DEFINE statement of MVS 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.

**Related Reading:** For a description of the use and format of passwords for VSAM, see *MVS Access Method Services*.

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 PASSWD=NO is specified and the data set is password protected at the control level (CONTROLPW) with passwords not equal to DBDNAME.

**NO**

Specifies that the DBDNAME for this DBD should not be used as the VSAM password. NO is the default.

**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 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. The following physical databases are supported by this exit routine:

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

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

**Related Reading:** For more information about this exit routine, see *IMS Version 8: Administration Guide: Database Manager*.

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 operand 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**

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

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

**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 created from saving physical segment data.
NODATA is optional.

**NOPATH**
Indicates 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 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 primary key. The DB2 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.
PATH is optional.

**CASCADE**
Indicates 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.

**Related Reading:** For a detailed discussion of delete rules for the Data Capture exit routine, see [IMS Version 8: Administration Guide: Database Manager].
CASCADE is the default.

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

NOKEY is optional.

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.

NODATA is optional.

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.

PATH is optional.

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

NOCASCADE is optional.

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

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

VERSION(character string)
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. If no parameter is specified, NO is implied.

YES
Specifies that 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.

NO
Allows the user exit, DFSDBUX1, to be called, provided DFSDBUX1 is located in the SDFSRESL. If DGSDBUX1 does not need to be called again for the DBD, X'FF' is returned in the SRCHFLAG field in the JCB, and DFSDLA00 dynamically marks the DBD as not requiring the exit. In this case, the user exit is not called again for that DBD for the duration of the IMS session, unless the DMB is purged from the DMB pool.

DATASET Statements

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

Requirement: At least one DATASET statement is required for each DBD generation.

Restriction: Data set statements are not allowed for HALDBs. Partitions are defined outside DBDGEN.

DEDB databases use AREA statements, not DATASET statements (see “AREA Statement” on page 44).

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 31.

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 32 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 32. 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/physical twin pointers, as opposed to hierarchic pointers, when they are in different data set groups, as shown in Figure 4.

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 via 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 via a physical child. The connection between multiple occurrences of segments B and D under one parent must be made via physical twin pointers.

Figure 4. Connections through Physical Child and 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 MVS 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 MVS assembler language statement.
- Unlabeled DATASET statements must have operands.

Referring to Figure 4 on page 31 illustrates use of the label field of the DATASET statement to group segment types of the same size in the same data set groups.

Table 3. Using the Label Field to Group Segment Types

<table>
<thead>
<tr>
<th>Label</th>
<th>Operation</th>
<th>Operand</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>DBD</td>
<td>NAME=HDBASE,ACCESS=HDAM,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RMNAME=(RANDMODL,1,500,824)</td>
</tr>
<tr>
<td>DSG1</td>
<td>DATASET</td>
<td>DD1=PRIMARY,BLOCK=1648</td>
</tr>
<tr>
<td></td>
<td>SEGM</td>
<td>NAME=SEGMENTA,BYTES=100</td>
</tr>
<tr>
<td>DSG2</td>
<td>DATASET</td>
<td>DD1=SECOND,BLOCK=3625</td>
</tr>
<tr>
<td></td>
<td>SEGM</td>
<td>NAME=SEGMENTB,BYTES=50,PARENT=SEGMENTA</td>
</tr>
<tr>
<td>DSG1</td>
<td>DATASET</td>
<td>NAME=SEGMENTC,BYTES=100,PARENT=SEGMENTA</td>
</tr>
<tr>
<td>DSG2</td>
<td>DATASET</td>
<td>NAME=SEGMENTD,BYTES=50,PARENT=SEGMENTC</td>
</tr>
<tr>
<td>N/A</td>
<td>DBDGEN</td>
<td>N/A</td>
</tr>
<tr>
<td>N/A</td>
<td>FINISH</td>
<td>N/A</td>
</tr>
<tr>
<td>N/A</td>
<td>END</td>
<td>N/A</td>
</tr>
</tbody>
</table>

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. The parameters are described in "DATASET Statement Parameter Description" on page 35.
DATASET Statement for HSAM Database

```
DATASET—DD1=ddname1, DD2=ddname2
, BLOCK=(blkfact1, blkfact2)  (1)
, RECORD=(reclen1, reclen2)  (1)
```

Notes:
1. If you do not specify a value, DBDGEN generates the value used.

DATASET Statement for GSAM Database

```
DATASET—DD1=ddname1
, DD2=ddname2
, BLOCK=(blkfact1)  (1)
, SIZE=size1  (1)
, RECORD=(reclen1, reclen2)  (1)
, RECFM=recfm1
```

Notes:
1. If you do not specify a value, DBDGEN generates the value used.
2. RECFM is only valid for a GSAM database.

DATASET Statement for HISAM Database

```
DATASET—DD1=ddname1, OVFLW=ddname3
, BLOCK=(blkfact1, blkfact2)  (2)
, SIZE=(size1, size2)  (2) (3)
, RECORD=(reclen1, reclen2)  (2)
```

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, DBDGEN generates the value used.
3. The valid parameter specifications for a SIZE keyword are 512 bytes, 1KB, 2KB, 4KB, 8KB, and multiples of 2KB up to 28KB.
DATASET Statement for HDAM Database

```
DATASET—DD1=ddname1
  ,BLOCK=size0
  ,SIZE=(,size1)
  ,SCAN=cyls
  ,FRSPC=(fbff,fspf)
  ,SEARCHA=0
```

Notes:
1. If you do not specify a value, DBDGEN generates the value used.
2. The valid parameter specifications for a SIZE keyword are 512 bytes, 1KB, 2KB, 4KB, 8KB, and multiples of 2KB up to 28KB. To ensure future compatibility, use only CI sizes that are multiples of 4KB.

DATASET Statement for HIDAM Database

```
DATASET—DD1=ddname1
  ,BLOCK=size0
  ,SIZE=(,size1)
  ,SCAN=cyls
  ,FRSPC=(fbff,fspf)
  ,SEARCHA=0
```

Notes:
1. If you do not specify a value, DBDGEN generates the value used.
2. The valid parameter specifications for a SIZE keyword are 512 bytes, 1KB, 2KB, 4KB, 8KB, and multiples of 2KB up to 28KB. To ensure future compatibility, use only CI sizes that are multiples of 4KB.

DATASET Statement for MSDB Database

```
DATASET—REL=(
  ,TERM
  ,FIXED
  ,DYNAMIC
)
```

DATASET Statement for INDEX Database

```
DATASET—DD1=ddname1
  ,OVFLW=ddname3
```

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, DBDGEN generates the value used.
3 The valid parameter specifications for a SIZE keyword are 512 bytes, 1KB, 2KB, 4KB, 8KB, and multiples of 2KB up to 28KB. To ensure future compatibility, use only CI sizes that are multiples of 4KB.

DATASET Statement for LOGICAL Database

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 operand must be specified if the ACCESS=LOGICAL operand 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 operand depends on the type of database being defined as shown in the following list:

<table>
<thead>
<tr>
<th>Database Type</th>
<th>Use of the DD1= operand</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSAM</td>
<td>DDname of input data set</td>
</tr>
<tr>
<td>GSAM</td>
<td>DDname of input data set</td>
</tr>
<tr>
<td>HISAM</td>
<td>DDname of primary data set in data set group</td>
</tr>
<tr>
<td>HIDAM</td>
<td>DDname of data set in data set group</td>
</tr>
<tr>
<td>HDAM</td>
<td>DDname of data set in data set group</td>
</tr>
<tr>
<td>MSDB</td>
<td>Operand is invalid</td>
</tr>
<tr>
<td>DEBB</td>
<td>Name of defined area</td>
</tr>
<tr>
<td>INDEX</td>
<td>DDname of primary data set</td>
</tr>
</tbody>
</table>
LOGICAL

Operand is invalid

For an HSAM 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 data set group 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 simple HSAM database and optional for a GSAM database. If it is omitted, DDname1 is assumed. This output data set is used by HSAM or GSAM when loading the database.

OVFLW=

Specifies the 1- to 8-character alphanumeric DDname of the overflow data set in this data set group. This operand 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 operand is not allowed when a simple HISAM database is defined.
- When a HISAM database that contains only one segment type is defined, the OVFLW operand does not have to be specified.
- No OVFLW operand 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.

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. Table 4 on page 37 explains the use of the BLOCK= and RECORD= operands.

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

In cases where the RECORD= and BLOCK= operands are used, the resulting control interval size must be a multiple of 512 when the resulting size is less than 8192 bytes. If the product of the record length specified times the blocking factor specified 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 32KB.

For HDAM and HIDAM databases, the BLOCK= operand 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= operand, 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= operand 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.

### Table 4. BLOCK= and RECORD= Operands

<table>
<thead>
<tr>
<th>Database Type</th>
<th>Use of BLOCK= and RECORD= Operands</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSAM</td>
<td>BLOCK=</td>
</tr>
<tr>
<td></td>
<td>blkfact1 applies to input data set and should always be 1.</td>
</tr>
<tr>
<td></td>
<td>blkfact2 applies to output data set and should always be 1.</td>
</tr>
<tr>
<td></td>
<td>RECORD=</td>
</tr>
<tr>
<td></td>
<td>reclen1 is the input record length.</td>
</tr>
<tr>
<td></td>
<td>reclen2 is the output record length.</td>
</tr>
<tr>
<td></td>
<td>HSAM is always unblocked; LRECL and BLKSIZE are equal.</td>
</tr>
<tr>
<td>GSAM</td>
<td>BLOCK=</td>
</tr>
<tr>
<td></td>
<td>blkfact1 applies to input/output data set.</td>
</tr>
<tr>
<td></td>
<td>blkfact2 is an invalid subparameter.</td>
</tr>
<tr>
<td></td>
<td>RECORD=</td>
</tr>
<tr>
<td></td>
<td>reclen1 is the size of an LRECL length or maximum size for a variable length record.</td>
</tr>
<tr>
<td></td>
<td>reclen2 is the minimum size for a variable length record.</td>
</tr>
<tr>
<td></td>
<td>SIZE=</td>
</tr>
<tr>
<td></td>
<td>size1 is the BLKSIZE for input/output data set.</td>
</tr>
<tr>
<td></td>
<td>size2 is an invalid subparameter.</td>
</tr>
<tr>
<td>HISAM</td>
<td>BLOCK=</td>
</tr>
<tr>
<td></td>
<td>blkfact1 is the primary data set blocking factor.</td>
</tr>
<tr>
<td></td>
<td>blkfact2 is the overflow data set blocking factor.</td>
</tr>
<tr>
<td></td>
<td>RECORD=</td>
</tr>
<tr>
<td></td>
<td>reclen 1 is the data set logical record length.</td>
</tr>
<tr>
<td></td>
<td>reclen2 is the overflow data set logical record length.</td>
</tr>
</tbody>
</table>
Table 4. BLOCK= and RECORD= Operands (continued)

<table>
<thead>
<tr>
<th>Database Type</th>
<th>Use of BLOCK= and RECORD= Operands</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIDAM HDAM</td>
<td>BLOCK= size0 is size without overhead of OSAM or VSAM data set group</td>
</tr>
<tr>
<td></td>
<td>RECORD= Is ignored.</td>
</tr>
<tr>
<td>MSDB</td>
<td>BLOCK= and RECORD= operands are invalid</td>
</tr>
<tr>
<td>DEDB</td>
<td>BLOCK= and RECORD= operands are invalid.</td>
</tr>
<tr>
<td>INDEX</td>
<td>BLOCK= blkfact1 is the primary data set blocking factor. blkfact2 is the overflow data set blocking factor.</td>
</tr>
<tr>
<td></td>
<td>RECORD= reclen1 is the primary data set logical record length. reclen2 is the overflow data set logical record length.</td>
</tr>
<tr>
<td>LOGICAL</td>
<td>BLOCK= and RECORD= operands are invalid.</td>
</tr>
</tbody>
</table>

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=
Is used to override DBDGEN's computation of 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 operand is used. The size1 operand 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 32767 bytes. An OSAM data set with an even length blocksize 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 32767. If the DBDGEN utility specifies the block size, 32752 bytes is the maximum amount specified.

Calculating SIZE= for HISAM Primary Data Set Groups, Primary HIDAM Index, 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 operand 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:
  – \( \text{size} \geq \text{MAXSEG} + (\text{RAPs} \times 4) + \text{FSEAP} + \text{VSAM CONTROL} \)

\text{RAPs=}
Number of root anchor points specified for the root addressable area of the database.

\text{FSEAP=}
4 bytes for a free space element anchor point.

\text{VSAM CONTROL=}
0 bytes for OSAM; 7 bytes for VSAM.

\text{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**

\( \text{size} \geq \text{MAXSEG} + \text{FSEAP} + \text{VSAM CONTROL} \)

\text{MAXSEG=}
Length in bytes of the longest segment in this data set group including the segment.

\text{FSEAP=}
4 bytes for a free space element anchor point.

\text{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 MVS 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:

\( \text{size} \geq \text{MAXSEG} + \text{FSEAP} + \text{RAP} + \text{VSAM CONTROL} \)

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

\( \text{size} \geq \text{MAXSEG} + \text{FSEAP} + \text{VSAM CONTROL} \)

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

\text{FSEAP=}
4 bytes for a free space element anchor point.

\text{VSAM CONTROL=}
0 bytes for OSAM; 7 bytes for VSAM.

\text{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 operand is optional and cannot be specified if ACCESS=LOGICAL is used on the DBD statement. reclen1 and reclen2 must be numeric values. The value of reclen2 must always be equal to or greater than the value of reclen1 except for GSAM databases. The meaning of each of the operand's parameters depends on the type of database being defined as shown in Table 4 on page 37. For a simple HISAM 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= operand, 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.

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 operand 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 operand 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 operand, 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 operand, 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.

**Related Reading:** Refer to [IMS Version 8: Administration Guide: Database Manager](#) for more information about the HD space search algorithm.

**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/ESA® 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.

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 an MVS 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 prime 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 DFSVSMPxx.

Related Reading: Refer to [IMS Version 8: Installation Volume 2: System Definition and Tailoring](#) for more information on the IMS.PROCLIB data set.
DD Statements

DD Statements Required for OSAM

Related Reading: Refer to the [IMS Version 8: Customization Guide](#) for Operating system procedures for execution of all IMS region types. The appropriate DD statements must be appended to these procedures.

For HSAM, 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 output data set, the data set must be preallocated, or the SPACE= operand 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= operand 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

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

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 DDB 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.

An example of the AREA statement follows. The parameters are explained in "AREA Statement Parameter Description" on page 45.

AREA Statement for DEDB Databases

```
(1)
```

```
|    | AREA—DD1=ddname1—,SIZE=size1—,UOW=(number1,overflow1) | AREA—ROOT=(number2,overflow2) |
```

Notes:

1. The valid parameter specifications for a DEDB SIZE keyword are 512 bytes, 1KB, 2KB, 4KB, 8KB and multiples of 4KB up to 28KB. To ensure future compatibility, use only C1 sizes that are multiples of 4KB.

### AREA Statement Parameter Description

<table>
<thead>
<tr>
<th>AREA</th>
<th>Identifies this statement as a DEDB AREA control statement.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DD1=</td>
<td>Specifies the DDname of the defined area. DDname1 must be a 1- to 8-character alphanumeric 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.</td>
</tr>
<tr>
<td>DEVICE=</td>
<td>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.</td>
</tr>
<tr>
<td>SIZE=</td>
<td>Specifies the control interval. Size can be 512 bytes, 1KB, 2KB, 4KB, and 8KB and multiples of 4KB up to 28KB. For future compatibility, only CI sizes that are multiples of 4KB should be used. No default value is allowed.</td>
</tr>
</tbody>
</table>

#### Restriction:

4KB 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.

<table>
<thead>
<tr>
<th>UOW=</th>
<th>Specifies the number of control intervals in a unit of work (UOW). The UOW= parameter has two operands, number1 and overflow1.</th>
</tr>
</thead>
<tbody>
<tr>
<td>number1</td>
<td>Specifies the number of control intervals in a unit of work (UOW). Its value must be from 2 to 32767.</td>
</tr>
<tr>
<td>overflow1</td>
<td>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.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>ROOT=</th>
<th>Specifies characteristics of a DEDB area. The ROOT= parameter has two operands, number2 and overflow2.</th>
</tr>
</thead>
<tbody>
<tr>
<td>number2</td>
<td>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.</td>
</tr>
</tbody>
</table>
**AREA Statement Parameter Descriptions**

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.

**Related Reading:** Refer to [IMS Version 8: Administration Guide: Database Manager](#) for more information about this procedure.

**SEGM Statement**

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 defined by the SEGM statement is placed in the data set group defined by the DATASET statement. Except for MSDBs and DEDBs, a maximum of 255 SEGM statements are allowed in a DBD generation. For a 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.

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. The parameters are explained in "SEGM Statement Parameter Description" on page 56.

**SEGM Statement for HSAM Databases**

```
SEGMENT NAME=segname1 PARENT=segname2,BYTES=max bytes
```
Notes:
1. The PARENT=keyword can be omitted, or PARENT=0 can be specified for the root segment type of a database.

SEGM Statement for HISAM Databases

SEGM NAME=segname1, PARENT=

(segname2,)

(VIRTUAL, dbname1)

(PHYSICAL, dbname1)

BYTES=max bytes

(max bytes, min bytes)

,FREQ=frequency

,POINTER=

,LPARENT,CTR,PAIRED

 RULES=

,LAST,FIRST,HERE

EXIT=

,NONE, A, C

,EXIT

,LOG, NOLOG

,LOG, NOLOG

,COMPRTN=

,routinename, DATA, KEY,

,INIT, max, PAD

A:

exitname

NOLOG

LOG

NOLOG

LOG

NOLOG
SEGM Statement

B:

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

C:

(8)

- CASCADE
- NOCASCADE
- (CASCADE B)
- (NOCASCADE B)

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. When using Fast Path sequential dependent segment processing, the insert rule of FIRST is always used and cannot be overridden.
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 with full-function and DEDB databases.
6 Variable-length segments and segment edit/compression cannot be specified for a simple HISAM database.
7 If an exit routine is not required because only logging is being requested, then if you specify the exit name as , the logging parameter defaults to LOG.
8 Used to control the CASCADE options.

SEGM Statement for HDAM Databases

- SEGM NAME=segname1, PARENT= 
- SNGL
- VIRTUAL
- PHYSICAL
- bytes=max bytes, min bytes
- HIER, LTWIN, LPARNT
- HIERBWD, TWIN, TWINBWD, NOTWIN
- CTR, PAIRED
**SEGM Statement**

1. **RULES:** 
   - FIRST
   - LAST

2. **EXIT:**
   - NONE
   - A
   - C

3. **COMPRTN:**
   - routinename
   - DATA
   - KEY
   - INIT
   - MAX
   - PAD

4. **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. 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 with full-function and DEDB databases.
   7. If an exit routine is not required because only logging is being requested, then if you specify the exit name as , the logging parameter defaults to LOG.
SEGM Statement

8 Used to control the CASCADE options.

SEGM Statement for PHDAM Databases

```plaintext
SEGM
NAME=segname1,PARENT=

((segname2
  SNGL
  DBLE
  lpsegname
  PHYSICAL
  dbname1)

BYTES=max bytes
(max bytes,min bytes)

POINTER=TWIN
PTR=TWINBWD
PAIRED

RULES=(P,P,P)
FIRST
HERE

DSGROUP={A,B,C,D,E,F,G,H,I,J}

EXIT=(NONE)

COMPRTN=(routinename
  DATA
  KEY
  INIT
  max)

A:

exitname
NOLOG
LOG
NOLOG
LOG
NOLOG
```
SEGM Statement

B:

- KEY
- NOKEY
- PATH
- NOPATH
- DATA
- NODATA

C:

(8)

- CASCADE
- NOCASCADE
- (CASCADE B)
- (NOCASTACE B)

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. 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 with full-function and DEDB databases.
7. If an exit routine is not required because only logging is being requested, then if you specify the exit name as , the logging parameter defaults to LOG.
8. Used to control the CASCADE options.

SEGM Statement for HIDAM Databases
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 with full-function and DEDB databases.

7 If an exit routine is not required because only logging is being requested, then if you specify the exit name as , the logging parameter defaults to LOG.

8 Used to control the CASCADE options.

SEGM Statement for PHIDAM Databases
SEGM Statement

A:

\[ \text{exitname, NOLOG, LOG, NOLOG} \]

B:

\[ \text{, KEY, PATH, DATA, NOKEY, NOPATH, NODATA} \]

C:

\[ \text{, CASCADE, KEY, (NOCASCADE, B), (NOCASCADE, B)} \]

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. When using Fast Path sequential dependent segment processing, the insert rule of FIRST is always used and cannot be overridden.
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 with full-function and DEDB databases.
8 If an exit routine is not required because only logging is being requested, then if you specify the exit name as , the logging parameter defaults to LOG.
9 Used to control the CASCADE options.

SEGM Statement for MSDB Databases

\[ \text{SEGM NAME=segname1,BYTES=max bytes} \]

SEGM Statement for DEDB Databases

\[ \text{SEGM NAME=segname1, PARENT=segname2, SNGL, DBLE} \]
The PARENT=keyword can be omitted, or PARENT=0 can be specified for the root segment type of a database.
SEGM Statement

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 with full-function and DEDB databases.

6 If an exit routine is not required because only logging is being requested, specify the exit name as * and the default logging parameter is LOG. If you do specify an exit routine name, the default logging parameter is NOLOG.

7 Used to control the CASCADE options.

SEGM Statement for INDEX Databases

```
>>>SEGM—NAME=segname1,BYTES=max bytes
```

(dchange)

\[FREQ=frequency\]

Notes:

dchange

FREQ=parameter will be ignored for PSINDEX

SEGM Statement for PSINDEX Databases

```
>>>SEGM—NAME=segname1,BYTES=max bytes
```

SEGM Statement Parameter Description

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

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>POINTER</td>
<td>PTR</td>
</tr>
<tr>
<td>FIRST</td>
<td>F</td>
</tr>
<tr>
<td>LAST</td>
<td>L</td>
</tr>
<tr>
<td>HERE</td>
<td>H</td>
</tr>
<tr>
<td>KEY</td>
<td>K</td>
</tr>
<tr>
<td>DATA</td>
<td>D</td>
</tr>
<tr>
<td>VIRTUAL</td>
<td>V</td>
</tr>
<tr>
<td>PHYSICAL</td>
<td>P</td>
</tr>
</tbody>
</table>

SEGMENT Identifies this statement as a segment definition statement.
NAME=
  Specifies the name of the segment type being defined. The specified name is
  used by DL/I and application programs in all references to this segment.
  Duplicate segment names are not allowed within a DBD generation. The
  segname1 operand must be a 1- to 8-character 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.

PARENT=
  Specifies the names of the physical and logical parents of the segment type
  being defined, if any. Optional but must have a value of 0 if used.

BYTES=
  Specifies the length of the data portion of a segment type in bytes using
  unsigned decimal integers. This parameter is required.

  **maxbytes and minbytes in fixed-length segments**
  For fixed-length segments, “maxbytes” specifies the amount of storage used
  for the data portion of the segment. The minbytes operand cannot be
  specified for a fixed-length segment. This includes a fixed-length
  compressed segment. The maximum length specified for a segment type
  must not exceed the maximum record length of the storage device used
  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 operand is
  included. The maxbytes field specifies the maximum length of any
  occurrence of this segment type. The maximum and minimum allowable
  values for the maxbytes operand are the same values as described for a
  fixed-length segment.

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

  • For a segment type that is not processed by an edit/compression routine
    or is processed by an edit/compression routine but the key compression
    option has not been specified, 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 or simple HISAM database cannot be
variable-length, the minbytes operand 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 minbyte
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=
Is only used for HSAM, HISAM, or INDEX databases. It specifies the estimated
number of times that this segment is likely to occur for each occurrence of its
physical parent. The frequency operand must be an unsigned decimal number
in the range 0.01 to $2^{24}-1$. If this is a root segment, “frequency” is the estimate
of the maximum number of database records that appear in the database being
defined. The value of the FREQ= operand when applied to dependent
segments is used to determine the logical record length and physical storage
block sizes for each data set group of the 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 occurs
during an HSAM or HISAM DBD generation, you can determine the logical
record length and physical block size.

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

The use of the POINTER= operand is primarily for HDAM, HIDAM, PHDAM,
and PHIDAM databases. In addition, it can be used for segment types 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 database, the
POINTER= operand must be omitted. If the segment type being defined is in a
HISAM database and does not participate in a logical relationship, the
POINTER= operand should be omitted.

The following abbreviations can be used in the pointer keyword options:

• HIER[H]
• HIERBWD[HB]
• TWIN[T]
• TWINBWD[TB]
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 (shown in brackets) can be selected.

The keyword options of the POINTER= operand 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.

**Related Reading:** Refer to [IMS Version 8: Administration Guide: Database Manager](#) for a more detailed explanation of the use of PTR=TWIN in HIDAM database root segments.

**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.

**Related Reading:** For more information on pointer fields, see [IMS Version 8: Administration Guide: Database Manager](#).

**NOTWIN [NT]**
Is used to prevent reserving space for a physical twin forward pointer in the prefix of occurrences of the segment type being defined. NOTWIN can be specified for a dependent segment type if the physical parent does not have hierarchic pointers specified, and no more than one occurrence of the dependent segment type 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 HIDAM and PHIDAM databases. When NOTWIN is specified for a dependent segment type and an attempt is made to load or insert a second occurrence of the dependent segment as a physical child of a given physical parent segment, an LB status code is returned when trying to insert the second occurrence during initial load, and an II status code is returned when trying to insert the second occurrence after initial load. The
NOTWIN option can be specified for HDAM and PHIDAM root segments but only when the randomizing module does not produce synonyms (keys with different values having the same block and anchor point). 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. It should be noted that 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. It should be noted that 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]**
Reserves a 4-byte logical parent pointer field in the prefix of occurrences of the segment type being defined. This parameter can only be specified when the segment type being defined is a logical child and the logical parent is in an HDAM or HIDAM database. If the logical parent is in a HISAM database, this parameter must be omitted, and the PARENT= operand for the segment being defined must specify PHYSICAL.

**CTR [C]**
Reserves a 4-byte counter field in the prefix of occurrences of the segment type being defined. A counter is required if a logical parent segment in a HISAM, HDAM, or HIDAM 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= Operand Default Values**
The default option for the POINTER= operand in any HDAM 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= operand of a SEGM statement.

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

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

If the POINTER= operand is explicitly stated on a SEGM statement, 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 operand is
omitted entirely. Table 5 illustrates use of the POINTER= operand parameters
for various types of DBD generations.

Table 5. Use of POINTER= Operand Parameters (No Logical Relationship)

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Keyword Parameter</th>
<th>Logical Segments GSAM MSDB DEDB</th>
<th>Physical Segments Contained in Database Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pointer to next segment in hierarchy</td>
<td>HIER</td>
<td>INVALID</td>
<td>HSAM SHSAM SHISAM HISAM HDAM HIDAM PHDAM PHIDAM INDEX PSINDEX</td>
</tr>
<tr>
<td>HIERBWD</td>
<td>INVALID</td>
<td>INVALID</td>
<td>Ign</td>
</tr>
<tr>
<td>TWIN</td>
<td>INVALID</td>
<td>INVALID</td>
<td>Ign</td>
</tr>
<tr>
<td>TWINBWD</td>
<td>INVALID</td>
<td>INVALID</td>
<td>Ign</td>
</tr>
<tr>
<td>CTR</td>
<td>INVALID</td>
<td>INVALID</td>
<td>VALID</td>
</tr>
<tr>
<td>LTWIN</td>
<td>INVALID</td>
<td>INVALID</td>
<td>Ign</td>
</tr>
<tr>
<td>LTWINBWD</td>
<td>INVALID</td>
<td>INVALID</td>
<td>Ign</td>
</tr>
<tr>
<td>LPARNT</td>
<td>INVALID</td>
<td>INVALID</td>
<td>VALID</td>
</tr>
<tr>
<td>PAIRED</td>
<td>INVALID</td>
<td>INVALID</td>
<td>VALID</td>
</tr>
</tbody>
</table>

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.

Notes:
1. Used when a logical child segment being defined participates in a logical relationship. VALID should be specified if the segment exists within HDAM,
   PHDAM, HIDAM, 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, PHDAM, HIDAM, or
   PHIDAM database.
3. Can be used when a logical child segment is being defined in an HDAM, PHDAM, HIDAM or PHIDAM database and the logical parent is in an
   HDAM, PHDAM, HIDAM, 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.

Related Reading: Refer to the appendix, “Replace, Insert, and Delete Rules for Logical Relationships,” in the IMS Version 8: Administration Guide: Database Manager for a description of the various uses of this keyword.

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 or LAST or 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.

In update mode, while processing HDAM and PHDAM roots without unique sequence fields, IMS sample randomizing modules (DFSHDC10 through DFSHDCC40) use the segment I/O area data to calculate a block/rap for an insert call.

Related Reading: For more information about HDAM Randomizing Routines, see IMS Version 8: Customization Guide.

The rules of FIRST, LAST, or HERE are only valid for update mode after a database has been loaded, except for the HDAM and PHDAM exceptions noted in Table 5 on page 61. 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:

```plaintext
SOURCE=((segname, DATA, dbname))
```

**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.
**SEGM Statement Parameter Descriptions**

**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:

```
SOURCE=-(segname, KEY, dbname), -(segname, DATA, dbname)
```

**(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 operand 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= operand 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.

**Related Reading:** For more information about this exit routine, see [IMS Version 8: Administration Guide: Database Manager](#).

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 operand 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.

**KEY**
Specifies 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.

NOKEY is optional.

**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.

NODATA is optional.

**NOPATH**
Indicates 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 primary key. The DB2 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.

PATH is optional.

**CASCADE**
Indicates 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.

**Related Reading:** For a detailed discussion of delete rules for the Data Capture exit routine, see [IMS Version 8: Administration Guide: Database Manager](#).
CASCADE is the default.

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

**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.

NOKEY is optional.

**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.

NODATA is optional.

**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.

PATH is optional.

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

NOCASCADE is optional.

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

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

**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, simple HSAM, simple HISAM, 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.

Note: 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 and must not be the same as any other name in IMS.SDFSRESL.

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 operand is selected.

KEY
Specifies that the exit routine can condense or modify any and all fields within the named segment. This parameter is invalid for the root segment of a HISAM 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 32767 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 MVS load module name of the user-supplied segment edit/compression exit routine.

Requirement: 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 operand is not supported for DEDB. If you specify the KEY operand, 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.

**Note:** The COMPRTN= keyword is prohibited on DEDB segments containing a unique key field located at the end of the segment. If COMPRTN= is attempted against these types of segments, DBDGEN fails and message DGEN440 is issued.

**Related Reading:** See [IMS Version 8: Messages and Codes, Volume 1](#) for a description of the DFEN440 message.

---

**LCHILD Statement**

The LCHILD statement defines the following:

- A logical relationship between two segment types in a HISAM, HIDAM, HDAM, PHDAM or PHIDAM database or a logical relationship between a segment type in any two of these databases
- A primary HIDAM index or secondary index relationship between two segment types
- No LCHILD statement should be entered 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= operand of a SEGM statement that defines a virtual logical child segment type establishes the same relationship between a logical parent and a virtual logical child segment type.

**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.

Following a SEGM statement that defines an index pointer segment type in a
secondary index database, there must be an LCHILD statement that identifies its
index target segment type.

A maximum of 255 LCHILD statements can occur in a single DBD generation. An
LCHILD statement 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 database, LCHILD statements are
invalid when ACCESS=HSAM.

The format of the LCHILD statement for each database type is shown in the
following examples. The parameters are explained in "LCHILD Statement
Parameter Description" on page 72.

**LCHILD Statement for HISAM Databases**

```
LCHILD (1)
  (2)
  NAME=(segname1,dbname)
  ,POINTER=DBLE
  ,PTR=
  (2)
  NONE
  (3)
  SYMB
  (2)
  ,PAIR=segname2
  ,PAIR=
  ,RULES=
  LAST
  FIRST
  HERE
```

**Notes:**
1. 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.

**LCHILD Statement for HDAM and PHDAM Databases**

```
LCHILD (1)
  (2)
  NAME=(segname1,dbname)
  ,POINTER=DBLE
  ,PTR=
  (2)
  NONE
  (3)
  SYMB
  (2)
  ,PAIR=segname2
  ,PAIR=
  ,RULES=
  FIRST
  HERE
```

**Notes:**
1. 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.
Notes:
1 Logical relationships or secondary indexing.
2 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.
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.

LCHILD Statement for HIDAM and PHIDAM Databases
Restriction: Do not enter an LCHILD statement for the primary index of a PHIDAM database.

Notes:
1 Logical relationships or secondary indexing.
2 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.
3 Used for HDAM, HISAM, and HIDAM 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.
LCHILD Statement

LCHILD Statement for INDEX Databases

```
LCHILD NAME=(segname1,dbname)

(1)

POINTER=SNGL

(2)

PTR=

(3)

INDEX=fldname
```

Notes:
1. Logical relationships or 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.

LCHILD Statement for PSINDEX Databases

```
LCHILD NAME=(segname1,dbname)

(1)

POINTER=SNGL

(2)

INDEX=fldname, RKSIZE=
```

Notes:
1. Logical relationships or secondary indexing.
2. Required for primary index of HIDAM database.

LCHILD Statement Parameter Description

The following abbreviations can be used in place of keywords specified in the macro definition:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>POINTER</td>
<td>PTR</td>
</tr>
<tr>
<td>FIRST</td>
<td>F</td>
</tr>
<tr>
<td>LAST</td>
<td>L</td>
</tr>
<tr>
<td>HERE</td>
<td>H</td>
</tr>
<tr>
<td>NAME=</td>
<td></td>
</tr>
</tbody>
</table>

The segname1 operand 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 operand 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 1- to 8-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.

**Restriction:** For PHDAM and PHIDAM databases, only the operands `INDX` and `NONE` are supported. All other operands are treated as if errors are present.

**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=` operand is LAST.

**NONE**

Should be used when the logical relationship from the logical parent to the logical child segment is not implemented or not implemented with direct address logical child pointers. In this case, the relationship from logical parent to logical child does not exist or is maintained by using physically paired segments. No pointer fields are reserved in the logical parent segment.

**INDX**

Is specified on the LCHILD statement in a HIDAM 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=` operand 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.

**Note:** If the target database is a HALDB, the index database must be defined as a HALDB index by use of the `PSINDEX` operand in the DBD statement `ACCESS` parameter.
LCHILD Statement Parameter Descriptions

**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 operand in the INDEX DBDGEN is to prevent reserving space in the prefix of index pointer segments for the 4-byte direct address index target segment pointer that is not used when the index pointer is symbolic.

**PAIR=**
Is specified `segname2` for bidirectional logical relationships only. The `segname2` operand is the name of the logical child segment that is, physically or virtually, paired with the logical child segment specified in `segname1`. The `segname2` operand must be a 1- to 8-character 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` operand 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.

**Note:** This parameter is not needed for a primary index of a PHIDAM database.

**RKSIZE=**
Specifies the root key size of the target segment.

**Important:** 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.

FIELD Statement

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. A maximum of 1000 fields can be defined for all segments in a DBD generation, and a maximum of 255 fields can be defined for any segment type. A unique sequence field must be defined for the root segment 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 will 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 format of the FIELD statement is for each database type is shown in the following examples. The parameters are explained in “FIELD Statement Parameter Description” on page 77.
FIELD Statement

FIELD Statement for HSAM Databases

\[\text{FIELD} \quad \text{NAME} = \langle \text{fldname1} \rangle, \text{BYTES} = \langle \text{bytes} \rangle, \text{START} = \langle \text{startpos} \rangle, \text{SEQ}, \text{M}, \text{C}\]

FIELD Statement for HISAM Databases

\((1)\)

\[\text{FIELD} \quad \text{NAME} = \langle \text{fldname1} \rangle, \text{BYTES} = \langle \text{bytes} \rangle, \text{SEQ}, \text{M}, \text{C}\]

\((2)\)

\[\text{systrelfldname}\]

\((3)\)

\[\text{TYPE} = \langle \text{X} \text{P} \rangle\]

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 for HDAM and PHDAM Databases

\[\text{FIELD} \quad \text{NAME} = \langle \text{fldname1} \rangle, \text{BYTES} = \langle \text{bytes} \rangle, \text{SEQ}, \text{M}, \text{C}\]

\[(1)\]

\[\text{systrelfldname}\]

\[(2)\]

\[\text{TYPE} = \langle \text{X} \text{P} \rangle\]

Notes:
1. A system related field used for secondary indexing.
2. The TYPE=parameter is ignored for fields with a systrelfldname.
FIELD Statement

**Notes:**
1. A system related field used for secondary indexing.
2. The TYPE=parameter is ignored for fields with a systrelfldname.

**FIELD Statement for MSDB Databases**

```
FIELD—NAME={fldname1,SEQ,U},BYTES=bytes
```

**FIELD Statement for DEDB Databases**

```
FIELD—NAME=fldname1,SEQ,U,BYTES=bytes,START=startpos
```

**FIELD Statement for Index Databases**

```
FIELD—NAME=fldname1,SEQ,U,BYTES=bytes,START=startpos
```

**FIELD Statement Parameter Description**

**NAME=fldname1**

Specifies the name of the field being defined within a segment type. The name specified can be referred to by an application program in a DL/I call SSA. Duplicate field names must not be defined for the same segment type. fldname1 must be a 1- to 8-character alphanumeric value.

**SEQ**

Identifies this field as a sequence field in the segment type. FIELD statements containing the keyword SEQ must be the first FIELD statements following a SEGM statement in a DBD generation input deck. If the sequence field of a real logical child segment consists of any part of the logical parent’s concatenated
key, you must specify the PHYSICAL parameter in the SEGMENT statement in order for the logical child to include the concatenated key of the logical parent with the logical child in storage.

As a general rule, a segment can have only one sequence field. However, in the case of virtually paired bidirectional logical relationships, multiple FIELD statements 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 real logical child’s physical parent followed by any intersection data). Since 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 fieldname1 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 a 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.

Restriction: SEQ cannot be specified for the sequential dependent segment.

U or M

Qualifies the type of sequence (SEQ) field 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 SEGMENT or LCHILD statement for that segment.
Recommendation: It is highly recommended that all segments which participate in a logical relationship have unique sequence fields. This includes physical and logical parents as well as physical and logical child segments. 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.

systrelfldname

Defines a system related field which can only be used for secondary indexing.

There are two types of system-related fields:

- All of or a portion of the concatenated key of an index source segment type defined by the preceding SEGM statement. The name for this type of system-related field can be up to 8 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.

Example: Assume the concatenated key shown in Table 6.

Table 6. Sample Concatenated Key for an Index Source Segment Type

<table>
<thead>
<tr>
<th>Root key (10 bytes)</th>
<th>Dependent key (3 bytes)</th>
<th>Dependent key (3 bytes)</th>
<th>Dependent key (3 bytes)</th>
</tr>
</thead>
</table>

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 5 and 6 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=25

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 8 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.
FIELD Statement Parameter Descriptions

On an XDFLD statement, a /CK field can be included in the list of fields specified for either the subsequence or DDATA fields or both of an index pointer segment. A /SX field can only be included in the list of fields specified for the subsequence field of index pointer segments.

BYTES=
Specifies the length of the field being defined in bytes. For fields other than system-related fields, BYTES must be a valid self-defining term whose value does not exceed 255. If a concatenated key or a portion of a concatenated key of an index source segment type is defined as a system-related field, the value specified can be greater than 255, but must not exceed the length of the concatenated key of the index source segment. The length of a /SX system-related field is always 4 bytes; therefore, when specified, the BYTES operand is disregarded. For the sequence field of an MSDB segment, BYTES must not exceed 240. For the sequence field of a DEDB segment, BYTES must not exceed the value of minbytes specified for the segment.

START=
Specifies the starting position (startpos) of the field being defined in terms of bytes relative to the beginning of the segment. Startpos must be a numeric term whose value does not exceed 32767. Startpos for the first byte of a segment is one. 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 a SEGM statement defines a logical child segment, the first n bytes of the segment type is the logical or physical parent's concatenated key. A field starting in position one would define all or a portion of this field. A field starting in position n+1 would start with intersection data.

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 operand. The startpos operand for the /SX system-related field is disregarded.

TYPE=
Specifies the type of data that is to be contained in this field. The value of the parameter specified for this operand indicates that one of the following types of data is contained in this field:

X      Hexadecimal data
P      Packed decimal data
C      Alphanumeric data or a combination of types of data
F      Binary fullword data
H      Binary halfword data

Parameters F and H are valid only for MSDB databases.

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 operand, except when the defined field is used with field sensitivity or is in an MSDB.

Types X, C, P, H, and F are valid in an MSDB, 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 systrefldname in the field statement is defined as either /SX or /CK, the TYPE= parameter is ignored and no type is set.

When sensitivity to a field has been defined, the field is filled with a value under these conditions:
• When the application program is not sensitive to this field on an insert call
• When:
  – The application program replaces a variable-length segment with a segment that is longer than the existing segment
  – This field is in the added portion of the segment
  – The application program is not sensitive to this field
• When the application program retrieves a variable-length segment that does not contain this field

The TYPE parameter determines the value to be used, as follows:

<table>
<thead>
<tr>
<th>TYPE</th>
<th>Value Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Binary zeros</td>
</tr>
<tr>
<td>P</td>
<td>Packed decimal zero</td>
</tr>
<tr>
<td>C</td>
<td>Blanks</td>
</tr>
</tbody>
</table>

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.
**XDFLD Statement**

Use the XDFLD statement only for secondary index relationships. Its purpose is to define the name of an indexed field that is associated to an index target segment type, identify the index source segment type, and identify the index source segment fields that are used in creating a secondary index. In addition, information regarding suppressing the creation of index pointer segments is provided through this control statement.

**Restriction:** This statement cannot be used to reference a segment in a DBD where ACCESS=INDEX, SHSAM, SHISAM, HSAM, MSDB, or DEDB has been specified.

A maximum of 32 XDFLD statements are allowed per SEGM statement. The number of XDFLD and FIELD statements combined must not exceed 255 per SEGM statement, and must not exceed 1000 per DBD generation.

One XDFLD statement is required for each secondary index relationship. It must appear in the DBD generation input deck for the indexed 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. The parameters are explained in "XDFLD Statement Parameter Description" on page 83.

### XDFLD Statement for HISAM Databases

```
XDFLD NAME=fldname,SEGMENT=segname,CONST=char
,,SRCH=list1,,SUBSEQ=list2,,ODATA=list3,,NULLVAL=value1,,EXTRTN=name1
```

**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.

### XDFLD Statement for HDAM Databases

```
XDFLD NAME=fldname,SEGMENT=segname,CONST=char
```
XDFLD Statement

Notes:
1. The combined length of the CONSTANT, SEARCH, and SUBSEQUENCE fields must not exceed 240 bytes.

XDFLD Statement for PHDAM Databases

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 operand is omitted, the index target segment type is assumed to be the index source segment.

CONST=
Specifies a character with which every index pointer segment in a particular secondary index is identified. This operand is optional. The purpose of this operand is to identify all index pointer segments associated with each secondary index when multiple secondary indexes reside in the same secondary index database. Char must be a 1-byte self-defining term.
Restriction: CONST is not supported for HALDBs. An error will occur if it is present for a HALDB.

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.

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 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 field values are concatenated in the index pointer segment subsequence field. This operand is optional.

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 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 field values are concatenated in the index pointer segment duplicate data field. This operand is optional.

NULLVAL=
Lets you suppress the creation of index pointer segments when the index source segment data used in the search field of an index pointer segment contains the specified value.

The value1 operand must be a 1-byte self-defining term (X'10', C'Z', 5, or B'00101101') or the words BLANK or ZERO. BLANK is equivalent to C' ' or X'40'. ZERO is equivalent to X'00' or 0, but not C'0'. If a packed decimal value is required, it must be specified as a hexadecimal term with a valid number digit and zone or sign digit (X'3F' for a packed positive 3 or X'9D' for negative 9).

No indexing is performed when each field of the index source segment specified in the SRCH= operand has the value of this operand in every byte. For example, if the NULLVAL=C'9' were specified, 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.

**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 operand (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 or not 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.

**DBDGEN, FINISH, and END Statements**
There are three additional utility control statements. Two are required (DBDGEN and END) and one is optional (FINISH).

The DBDGEN statement indicates the end of DBD generation control statements used to define the DBD. This statement is required. The following example shows the format of the DBDGEN statement for all database types.

```
DBDGEN

```

The FINISH statement is optional and is retained for compatibility. The following example shows the format of the FINISH statement for all database types.

```
FINISH

```

The END statement indicates the end of input statements to the MVS assembler. This statement is required. The following example shows the format of the END statement for all database types.

```
END

```

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

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

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

**Assembler Listing**

An MVS 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.

**Note:** RECORDSIZE needs to be changed appropriately for all ESDS definitions.

If the control interval size is not specified (see SIZE= parameter), 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 MVS Access Method Services control statements. The first DEFINE
provides parameters for the key sequenced data set (KSDS) and the second
DEFINE provides parameters for the entry sequenced data set (ESDS).

To provide a complete definition for a VSAM data set, 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.

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. See Figure 5 on page 87 for an example of MVS Access Method
Services parameters from DBD generation.
Figure 6 shows the DBDGEN input used to create the output in Figure 5.

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 Figure 7 on page 88.
Segment Prefix Format Description

<table>
<thead>
<tr>
<th>CONVERTED BYTE</th>
<th>VALUE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td><strong>POINTER POSITIONS GENERATED:</strong></td>
</tr>
<tr>
<td>1...</td>
<td>CTR (Counter)</td>
<td></td>
</tr>
<tr>
<td>.1...</td>
<td>Physical twin forward</td>
<td></td>
</tr>
<tr>
<td>.11...</td>
<td>Physical twin forward and backward</td>
<td></td>
</tr>
<tr>
<td>....1...</td>
<td>Physical parent</td>
<td></td>
</tr>
<tr>
<td>....11...</td>
<td>Logical twin forward</td>
<td></td>
</tr>
<tr>
<td>....111...</td>
<td>Logical twin forward and backward</td>
<td></td>
</tr>
<tr>
<td>....1111...</td>
<td>Logical parent</td>
<td></td>
</tr>
<tr>
<td>....11111...</td>
<td>Hierarchic forward</td>
<td></td>
</tr>
<tr>
<td>....111111...</td>
<td>Hierarchic forward and backward</td>
<td></td>
</tr>
</tbody>
</table>

| 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** |

**Figure 7. Segment Flag Codes**

Segment Prefix Format Description

Output from DBD generation contains the statement:

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

Convert the values to binary and decimal representations:

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>FE</td>
<td>FD</td>
<td>08</td>
<td>0A</td>
</tr>
<tr>
<td>11111110</td>
<td>11111101</td>
<td>00000100</td>
<td>10</td>
</tr>
</tbody>
</table>

**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.
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 link-edit of the object module, which produces a load module that becomes a member of the IMS.DBDLIB library.

DBD Generation Error Conditions

Related Reading: Refer to [IMS Version 8: Messages and Codes, Volume 1] for the DBD generation error messages.

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.

Examples of DBD Generation

This section contains examples of DBD generation for different database types.

Examples without Secondary Index or Logical Relationships

The DBD generation examples provided in the following section show the statements that are required to define HSAM, HISAM, HDAM, HIDAM, primary HIDAM Index, GSAM, and MSDB and DEDB databases without secondary indexes or logical relationships. Two data structures are shown in Figure 8 on page 90. 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. One or both structures are the basis for the examples in Figure 8 on page 90 through Figure 19 on page 100.
Example of HSAM DBD Generation

The examples in Figure 9 show the DBD generation control 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

SEG  NAME=SKILL,BYTES=31,FREQ=100
     FIELD NAME=TYPE,BYTES=21,START=1,TYPE=C
     FIELD NAME=STDCODE,BYTES=10,START=22,TYPE=C

SEG  NAME=NAME,BYTES=20,FREQ=500,PARENT=SKILL
     FIELD NAME=STDCLEVL,BYTES=20,START=1,TYPE=C

SEG  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

SEG  NAME=EDUC,BYTES=75,FREQ=5,PARENT=NAME
     FIELD NAME=GRADLEVEL,BYTES=10,START=1,TYPE=C
     FIELD NAME=SCHOOL,BYTES=65,START=11,TYPE=C

DBDGEN
FINISH
END
```

*Figure 9. HSAM DBD Generation (Part 1 of 2)*
HSAM DBD Generation of Payroll Database

```
DBD NAME=PAYROLD,ACCESS=HSAM
DATASET DD1=PAYROLL,DD2=PAYOUT,BLOCK=1,RECORD=1000,

SEG M NAME=NAME,BYTES=150,FREQ=1000,PARENT=0
FIELD NAME=(EMPLOYEE,SEQ,U),BYTES=60,START=1,TYP= Type=C
FIELD NAME=MANNBR,BYTES=15,START=61,TYP= Type=C
FIELD NAME=ADDR,BYTES=75,START=76,TYP= Type=C

SEG M NAME=ADDRESS,BYTES=200,FREQ=2,PARENT=NAME
FIELD NAME=HOMEADDR,BYTES=100,START=1,TYP= Type=C
FIELD NAME=COMAILOC,BYTES=100,START=101,TYP= Type=C

SEG M NAME=PAYROLL,BYTES=100,FREQ=1,PARENT=NAME
FIELD NAME=HOURS,BYTES=15,START=51,TYP= Type=P
FIELD NAME=BASICPAY,BYTES=15,START=1,TYP= Type=P

DBDGEN
FINISH
END
```

Figure 9. HSAM DBD Generation (Part 2 of 2)

Example of HISAM DBD Generation

The examples in Figure 10 show the DBD generation control 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,

SEG M NAME=SKILL,BYTES=31,FREQ=100
FIELD NAME=(TYPE,SEQ,U),BYTES=21,START=1,TYP= Type=C
FIELD NAME=STDCODE,BYTES=10,START=22,TYP= Type=C

SEG M NAME=NAME,BYTES=20,FREQ=500,PARENT=SKILL
FIELD NAME=(STDCLEVL,SEQ,U),BYTES=20,START=1,TYP= Type=C

SEG M NAME=EXPR,BYTES=20,FREQ=10,PARENT=NAME
FIELD NAME=PREVJOB,BYTES=10,START=1,TYP= Type=C
FIELD NAME=CLASSIF,BYTES=10,START=11,TYP= Type=C

SEG M NAME=EDUC,BYTES=75,FREQ=5,PARENT=NAME
FIELD NAME=GRADLEVL,BYTES=10,START=1,TYP= Type=C
FIELD NAME=SCHOOL,BYTES=65,START=11,TYP= Type=C

DBDGEN
FINISH
END
```

Figure 10. HISAM DBD Generations (Part 1 of 2)
Examples

**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

*Figure 10. HISAM DBD Generations (Part 2 of 2)*

**Example of HDAM DBD Generation**

The examples in [Figure 11](#) show the control 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=(RAMMODL,1,500,824)
DATASET DD1=SKILHDAM,BLOCK=1648,SCAN=5

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

*Figure 11. HDAM DBD Generation (Part 1 of 3)*
**HDAM DBD Generation of Skills Inventory Database with Physical Child and Physical Twin Pointers**

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

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
FIELD NAME=CLASSIF,BYTES=10,START=11,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
```

*Figure 11. HDAM DBD Generation (Part 2 of 3)*

**HDAM DBD Generation of Skills Inventory SKILLINV Database with EXIT= and VERSION= Parameters**

```plaintext
DBD NAME=SKILLINV,ACCESS=HDAM,RMNAME=(RAMMODL,1,500,824),VERSION=CCCCCC
DATASET DD1=SKILHDAM,BLOCK=1648,SCAN=5

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
FIELD NAME=CLASSIF,BYTES=10,START=11,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
```

*Figure 11. HDAM DBD Generation (Part 3 of 3)*

**Example of HIDAM DBD Generation**

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. Figure 12 summarizes the statements required to establish the index relationship between the HIDAM root segment type and the
index segment type in the primary HIDAM index database. Only those operands pertinent to the index relationship are shown.

**Primary HIDAM Index Relationship**

<table>
<thead>
<tr>
<th>HIDAM:</th>
<th>INDEX:</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBD NAME=bd1,ACCESS=HIDAM</td>
<td>DBD NAME=bd2,ACCESS=INDEX</td>
</tr>
<tr>
<td>SEGM NAME=seg1,BYTES=,POINTER=</td>
<td>SEGM NAME=seg2,BYTES=</td>
</tr>
<tr>
<td>LCHILD NAME=(seg2,bd2),PTR=INDX</td>
<td>LCHILD NAME=(seg1,bd1),INDEX=fld1</td>
</tr>
<tr>
<td>FIELD NAME=(fld1,SEQ,U),BYTES=,START=</td>
<td>FIELD NAME=(fld2,SEQ,U),BYTES=,START=</td>
</tr>
</tbody>
</table>

*Figure 12. Summary of Statements for the Primary HIDAM Index Relationship*

The next two examples show the control statements that define the skills inventory data structure as two HIDAM 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. *Figure 13* shows the control statements for the two HIDAM DBD generations and the index DBD generation.

**INDEX DBD Generation for HIDAM Database SKILLINV**

```plaintext
DBD NAME=INDEXDB,ACCESS=INDEX
DATASET DD1=INDXDB1,
SEGM NAME=INDEX,BYTES=21,FREQ=10000
LCHILD NAME=(SKILL,SKILLINV),INDEX=TYPE
FIELD NAME=(INDEXSEQ,SEQ,U),BYTES=21,START=1
DBDGEN
FINISH
END
```

*Figure 13. HIDAM and Primary HIDAM Index DBD Generations (Part 1 of 3)*
HIDAM DBD Generation of Skills Inventory Database with Hierarchic Pointers

```plaintext
DBD  NAME=SKILLINV,ACCESS=HIDAM
DATASET DD1=SKLHIDAM,BLOCK=1648,SCAN=5

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=INDEX

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
```

Figure 13. HiDAM and Primary HiDAM Index DBD Generations (Part 2 of 3)

HIDAM DBD Generation of Skills Inventory SKILLINV Database with Physical Child and Physical Twin Pointers

```plaintext
DBD  NAME=SKILLINV,ACCESS=HIDAM
DATASET DD1=SKLHIDAM,BLOCK=1648,SCAN=5

SEGm  NAME=SKILL,BYTES=31,PTR=T,PARENT=0
LCHILD NAME=(INDEX,INDEXDB),PTR=INDEX

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
```

Figure 13. HiDAM and Primary HiDAM Index DBD Generations (Part 3 of 3)

Example of PHDAM DBD Generation

Figure 14 shows the DBD generation of skills inventory database with physical child and physical twin pointers for a PHDAM database.
Examples

**Example of PHIDAM DBD Generation**

Figure 15 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,OSAM),RMNAME=(RAMDMODL,1,500,824)
SEGMENT 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
SEGMENT NAME=NAME,BYTES=20,PTR=T,PARENT=((SKILL,SNGL))
FIELD NAME=(STDCLEVL,SEQ,U),BYTES=20,START=1,TYPE=C
SEGMENT 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
SEGMENT 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
```

*Figure 14. PHIDAM DBD Generations*

**Example of GSAM DBD Generation**

Figure 16 shows the DBD generation control 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
```

*Figure 15. PHIDAM DBD Generations*

**Example of MSDB DBD Generation**

Figure 17 on page 97 shows 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

Figure 17. Main Storage Database DBD Generations (Part 1 of 4)

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

Figure 17. Main Storage Database DBD Generations (Part 2 of 4)

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

Figure 17. Main Storage Database DBD Generations (Part 3 of 4)
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

Figure 17. Main Storage Database DBD Generations (Part 4 of 4)

Example of DEDB DBD Generation

Figure 18 on page 99 shows the DBD generation statements necessary to define a data entry database DBD.
DBD Generation for a DEDB

| DEDB1 | DBD | NAME=DEDB0001,ACCESS=DEDB,RMNAME=RMOD1 |
| AREA0 | AREA | D01=DB1AREA0, AREA 0 |
|       |       | MODEL=1,SIZE=1024, ROOT=(10,5), UOW=(15,10) 5 A.P.'S + 10 DEP. OFLOW. |
| AREA1 | AREA | D01=DB1AREA1, AREA 1 |
|       |       | MODEL=11,SIZE=1024, ROOT=(10,5), UOW=(15,10) 5 A.P.'S + 10 DEP. OFLOW. |
| AREA2 | AREA | D01=DB1AREA2, AREA 2 |
|       |       | SIZE=1024, ROOT=(10,5), UOW=(15,10) 5 A.P.'S + 10 DEP. OFLOW. |
| AREA3 | AREA | D01=DB1AREA3, AREA 3 |
|       |       | SIZE=4096, ROOT=(10,5), UOW=(15,10) 5 A.P.'S + 10 DEP. OFLOW. |
| AREA4 | AREA | D01=DB1AREA4, AREA 4 |
|       |       | MODEL=1,SIZE=2048, ROOT=(10,5), UOW=(15,10) 5 A.P.'S + 10 DEP. OFLOW. |
| AREA5 | AREA | D01=DB1AREA5, AREA 5 |
|       |       | MODEL=2,SIZE=4096, ROOT=(10,5), UOW=(15,10) 5 A.P.'S + 10 DEP. OFLOW. |
| AREA6 | AREA | D01=DB1AREA6, AREA 6 |
|       |       | SIZE=1024, ROOT=(10,5), UOW=(15,10) 5 A.P.'S + 10 DEP. OFLOW. |
| AREA7 | AREA | D01=DB1AREA7, AREA 7 |
|       |       | SIZE=2048, ROOT=(10,5), 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,TYP=C |
| SDSEG | SEGM | NAME=SDSEGM1,PARENT=ROOTSEG1,BYTES=(300,50), TYPE=SEQ |
| SDFLD | FIELD | NAME=SDSCFLD1,BYTES=10,START=3,TYP=C |
| DDSEG | SEGM | NAME=DDSEGM1,PARENT=ROOTSEG1, BYTES=(40,15),TYP=DIR |
| DDFLD1 | FIELD | NAME=(DD1FLD1,SEQ,U),BYTES=4,START=6 |
| DDFLD2 | FIELD | NAME=DD2FLD2,BYTES=5,START=10,TYP=P |

Figure 18. Data Entry Database DBD Generations

Figure 19 shows the DBD generation statements necessary to define a DEDB with subset pointers.
Examples

DBD Generation for DEDB Subset Pointers

DBD NAME=DEDBDB,ACCESS=DEDB,RMNAME=DBFHD040
AREA   DD1=DEDBDB,MODEL=1,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
FIELD  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

Figure 19. DBD Generation of DEDB Subset Pointers Sample

Note: SSPTTr=n, where n indicates the number of subset pointers

Summary of Physical Database Description Examples

An application program through a database PCB can operate on any of the
databases previously described. The value of the DBDNAME= operand on the
database PCB control statement should equal the value of the NAME= operand on
a DBD PCB statement of DBD generation. The SENSEG statements following the
database PCB 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= operand on the PCB statement should equal the value of the NAME=
operand on the DBD statement for the HIDAM DBD generation. The LCHILD
statement in the HIDAM DBD provides IMS with the relationship to the necessary
INDEX DBD and index database. The INDEX DBD name should not be specified in
the DBDNAME= operand of a database PCB.

Examples with Logical Relationships

Figure 20 on page 101 shows the three types of logical relationships (unidirectional,
bidirectional physically paired, and bidirectional virtually paired) that can be defined
in IMS databases. Also in the figure are the statements required to define each type
of relationship. Only the operands pertinent to the relationship are shown, and it is
assumed that each type of relationship is defined between segments in two
databases named DBD1 and DBD2.
Figure 20. Comparison of Unidirectional, Physically Paired Bidirectional, and Virtually Paired Bidirectional Logical Relationships (Part 1 of 2)
### Unidirectional Logical Relationships

<table>
<thead>
<tr>
<th>Statements for DBD1</th>
<th>Statements for DBD2</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEGMENT SEG1, PARENT= SEG2, BYTES=FREQ, POINTER= RULES=</td>
<td>SEGMENT SEG3, PARENT= SEG2, BYTES=FREQ, POINTER= RULES=</td>
</tr>
</tbody>
</table>

**Note:**
1. Specify symbolic and/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.

### Physically Paired Bidirectional Logical Relationships

<table>
<thead>
<tr>
<th>Statements for DBD1</th>
<th>Statements for DBD2</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEGMENT SEG1, PARENT= SEG2, BYTES=FREQ, POINTER= RULES=</td>
<td>SEGMENT SEG3, PARENT= SEG2, BYTES=FREQ, POINTER= RULES=</td>
</tr>
</tbody>
</table>

**Note:**
1. Specify symbolic and/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.

### Virtually Paired Bidirectional Logical Relationship

<table>
<thead>
<tr>
<th>Statements for DBD1</th>
<th>Statements for DBD2</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEGMENT SEG1, PARENT= SEG2, BYTES=FREQ, POINTER= RULES=</td>
<td>SEGMENT SEG3, PARENT= SEG2, BYTES=FREQ, POINTER= RULES=</td>
</tr>
</tbody>
</table>

**Notes:**
1. Specify symbolic and/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= operand 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.

---

*Figure 20. Comparison of Unidirectional, Physically Paired Bidirectional, and Virtually Paired Bidirectional Logical Relationships (Part 2 of 2)*

In the Virtually Paired Bidirectional Logical Relationship part of Figure 20 on page 101, 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.

Figure 21 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 Figure 21.

Part 1: Physical Databases

Payroll Database

```
NAMEMAST
  ADDRESS
  PAYROLL
```

Skills Inventory Database

```
SKILMAST
  SKILNAME
    EXPR
    EDUC
```

Part 2: Logical Relationship

Payroll Database

```
NAMEMAST
  NAMESKIL
  ADDRESS
  PAYROLL
```

Skills Inventory Database

```
SKILMAST
  SKILNAME
    EXPR
    EDUC
```

Part 3: Logical Databases

```
SKILL
  NAME
    ADDRESS
    PAYROLL
    SKILL
```

Figure 21. Logical Relationship Between Physical Databases and The Resulting Logical Databases That Can Be Defined

Figure 22 on page 104 shows the DBD generation statements necessary to define:
The payroll and skills inventory data structures depicted in Part 2 of Figure 21 as a HIDAM and HDAM data base with a virtually paired bidirectional logical relationship between the two databases.

The logical data structures depicted in Part 3 of Figure 21 as logical databases.

---

**Examples**

- The payroll and skills inventory data structures depicted in Part 2 of Figure 21 as a HIDAM and HDAM data base with a virtually paired bidirectional logical relationship between the two databases.

**DBD**

```
DBD NAME=PAYROLDB,ACCESS=HIDAM
DATASET DD1=PAYHIDAM,BLOCK=1648,SCAN=3
SEGMENT NAME=NAMEMAST,PTR=TWINBWD,RULES=(VVV), X
   BYTES=150
   LCHILD NAME=(INDEX,INDEXDB),PTR=INDEX
   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
   SEGMENT NAME=NAMESKIL,PARENT=NAMEMAST,PTR=PAIRED,
      SOURCE=((SKILNAME,DATA,SKILLINV))
   FIELD NAME=(TYPE,SEQ,U),BYTES=21,START=1,TYPE=C
   FIELD NAME=STDLEVL,BYTES=20,START=22,TYPE=C
   SEGMENT 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
   SEGMENT 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**

```
DBD NAME=SKILLINV,ACCESS=HDAM,RMNAME=(RAMDMODI,1,500,824)
DATASET DD1=SKILHDAM,BLOCK=1648,SCAN=5
SEGMENT NAME=SKILMAST,BYTES=31,PTR=TWINBWD
   FIELD NAME=(TYPE,SEQ,U),BYTES=21,START=1,TYPE=C
   FIELD NAME=STDLEVL,BYTES=20,START=22,TYPE=C
   SEGMENT NAME=SKILNAME,
      PARENT=((SKILMAST,DBLE),(NAMEMAST,P,PAYROLDB)), X
      BYTES=80,PTR=(LINDEX,LINDEXDB,TWINBWD), X
      RULES=(VVV)
   FIELD NAME=(EMPLOYEE,SEQ,U),START=1,BYTES=60,TYPE=C
   FIELD NAME=(STDLEVL),BYTES=20,START=61,TYPE=C
   SEGMENT NAME=EXPR,BYTES=20,PTR=T,
      PARENT=((SKILNAME,SNGL))
   FIELD NAME=PREVJOB,BYTES=10,START=1,TYPE=C
   FIELD NAME=CLASSIF,BYTES=10,START=11,TYPE=C
   SEGMENT NAME=EDUC,BYTES=75,PTR=T,
      PARENT=((SKILNAME,SNGL))
   FIELD NAME=GRADLEVL,BYTES=10,START=1,TYPE=C
   FIELD NAME=SCHOOL,BYTES=65,START=11,TYPE=C
DBDGEN FINISH END
```

*Figure 22. DBD Generation Statements Examples (Part 1 of 2)*
Examples with Secondary Indexes

The statements required to establish a secondary index relationship between a segment type in an indexed database and a segment type in a secondary index database are summarized in Table 7, Table 8 on page 106, and Table 9 on page 107. The statements required when the index target and index source segment types are the same are shown in Table 7. In Table 8 on page 106, the index target and index source segment types are different. Table 9 on page 107 shows the statements required for a shared secondary index DBD generation. In all three tables, only those operands pertinent to the secondary index relationships are shown.

Table 7. Same Index Source and Target Segment Types

<table>
<thead>
<tr>
<th>Indexed DBD</th>
<th>Index DBD</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBDNAME=DBD1, ACCESS=</td>
<td>DBDNAME=DBD2, ACCESS=INDEX</td>
</tr>
<tr>
<td>SEGMNAME=SEG1, PARENT=</td>
<td>SEGNAME=SEG3, PARENT=0, BYTES=</td>
</tr>
<tr>
<td>FIELDNAME=(FLD2, SEQ,...), BYTES=</td>
<td>FIELDNAME=(FLD2, SEQ,...), BYTES=</td>
</tr>
<tr>
<td>LCHILDNAME=(SEG3, DBD2), POINTER=INDEX</td>
<td>LCHILDNAME=(SEG1, DBD1), INDEX=XFLD, POINTER=SNGL</td>
</tr>
<tr>
<td>XDFLDNAME=XFLD, SRCH=FL</td>
<td></td>
</tr>
</tbody>
</table>

Notes to Table 7:

Figure 22. DBD Generation Statements Examples (Part 2 of 2)
Examples

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.

Table 8. Different Index Source and Target Segment Types

<table>
<thead>
<tr>
<th>Indexed DBD</th>
<th>Index DBD</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBDNAME=DBD1,ACCESS=</td>
<td>DBDNAME=DBD2,ACCESS=INDEX</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>SEGNAME1=SEG1,BYTES=,PARENT=</td>
<td>SEGNAME=SEG4,PARENT=0,BYTES=</td>
</tr>
<tr>
<td>LCHILDNAME=(SEG4,DBD2), POINTER2=INDEX</td>
<td>FIELDNAME=(FLD4,SEQ,...)</td>
</tr>
<tr>
<td>XLFLDNAME=XFLD,SEGMENT=SEG3, SRCH=FLD3,...</td>
<td>LCHILDNAME=(SEG1,DBD1), INDEX=XFLD,POINTER2=SNGL</td>
</tr>
<tr>
<td>SEGNAME=SEG2,BYTES=,PARENT=SEG1</td>
<td></td>
</tr>
<tr>
<td>SEGNAME1=SEG3 ,PARENT=SEG2</td>
<td></td>
</tr>
<tr>
<td>FIELDNAME=FLD3,BYTES= ,START=</td>
<td></td>
</tr>
</tbody>
</table>

Notes to Table 8

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.
Table 9. Shared Secondary Index Database DBD Generation

<table>
<thead>
<tr>
<th>Indexed DBD</th>
<th>Index DBD</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBDNAME=DBD1,ACCESS=</td>
<td>DBDNAME=(DBD2, DBD3), ACCESS=INDEX</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>SEGNAME=SEG1, BYTES=PARENT=</td>
<td>SEGNAME=SEG3, PARENT=0, BYTES=1</td>
</tr>
<tr>
<td>FIELDNAME=FLD1,BYTES=</td>
<td>FIELDNAME=FLD3, SEQ,...),</td>
</tr>
<tr>
<td>\hspace{1cm}START=</td>
<td>START=1, BYTES=</td>
</tr>
<tr>
<td>FIELDNAME=FLD2,BYTES=</td>
<td>LCHILDNAME=SEG1, DBD1),</td>
</tr>
<tr>
<td>\hspace{1cm}START=</td>
<td>INDEX=XFLD1</td>
</tr>
<tr>
<td>LCHILDNAME=(SEG3, DBD2), POINT=INDX</td>
<td>SEGNAME=SEG5, PARENT=0, BYTES=</td>
</tr>
<tr>
<td>XDLDNAME=XFLD1, SRCH=FLD2, CON=\textquoteleft C'2\textquoteright...</td>
<td>FIELDNAME=FLD10, SEQ,...),</td>
</tr>
<tr>
<td>\hspace{1cm}</td>
<td>\hspace{1cm}</td>
</tr>
<tr>
<td>\hspace{1cm}</td>
<td>START=1, BYTES=</td>
</tr>
<tr>
<td>SEGNAME=SEG2,BYTES=PARENT=</td>
<td>LCHILDNAME=(SEG2, DBD1),</td>
</tr>
<tr>
<td>FIELDNAME=FLD4,BYTES=</td>
<td>INDEX=XFLD2</td>
</tr>
<tr>
<td>\hspace{1cm}START=</td>
<td>\hspace{1cm}</td>
</tr>
<tr>
<td>LCHILDNAME=(SEG5, DBD3), POINT=INDX</td>
<td>\hspace{1cm}</td>
</tr>
<tr>
<td>XDLDNAME=XFLD2, SRCH=FLD4, CON=\textquoteleft C'1\textquoteright...</td>
<td>\hspace{1cm}</td>
</tr>
</tbody>
</table>

This example is shown with direct pointers for the index pointer segment types, and with the index source segment type, and the index target segment type the same. Symbolic pointing or differing index source and target segments types can be used; however, all secondary index databases in the shared index must uniformly specify either symbolic pointers or direct pointers; a mixture of symbolic and direct pointing is not allowed in a shared secondary index database.

Example of DBDGEN for Secondary Index Databases

Figure 23 on page 108 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 23. Database Indexed by Two Secondary Indexes

Figure 24 shows the DBD generation statements that define the indexed database and the secondary index databases.

**DBDGEN for Indexed Database**

```plaintext
DBDGEN
DBDNAME=DTA1,ACCESS=HDAM,RMNAME=(RANDMODL,1,500,824)
DATASET DD1=D1,MODEL=1
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
```

Figure 24. Indexed Database and Secondary Index Database (Part 1 of 3)
Example of DBDGEN for a Shared Secondary Index Database

Figure 25 shows a database, DTA3, that is indexed by three secondary indexes (X4, X5, and X6) in a shared secondary index database. 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
Figure 26 shows the DBD generation statements that define the indexed database, the primary index data base, and the shared secondary index database.

### DDBGEN for Indexed Database

```dbddgen
DBD NAME=DTA3,ACCESS=HIDAM
DATASET DD1=D1
SEGM NAME=DA,PARENT=0,BYTES=15
LCHILD NAME=(INDEX,X2),PTR=INDX
FIELD NAME=(DAF1,SEQ),BYTES=5,START=1
LCHILD NAME=(X4A,X4),PTR=INDX
XDFLD NAME=DAF1X,SRCH=DAF1,CONST=C'1'
SEGM NAME=DB,PARENT=DA,BYTES=20
FIELD NAME=(DBF1,SEQ),BYTES=5,START=1
SEGM NAME=DC,PARENT=DA,BYTES=20
FIELD NAME=(DCF1,SEQ),BYTES=5,START=1
LCHILD NAME=(X5A,X5),PTR=INDX
XDFLD NAME=DCF1X,SRCH=DCF1,CONST=C'2'
SEGM NAME=DE,PARENT=DC,BYTES=25
FIELD NAME=(DEF1,SEQ),BYTES=5,START=1
LCHILD NAME=(X6A,X6),PTR=INDX
XDFLD NAME=DEF1X,SRCH=DEF1,CONST=C'3'
DBDGEN
FINISH
END
```

### DDBGEN for Primary Index Database

```dbddgen
DBD NAME=X2,ACCESS=INDEX
DATASET DD1=X2P
SEGM NAME=INDEX,BYTES=5
LCHILD NAME=(DA,DTA3),INDEX=DAF1
FIELD NAME=(INDESEQ,SEQ,U),BYTES=5,START=1
DBDGEN
FINISH
END
```

### DDBGEN for Shared Secondary Index Database

```dbddgen
DBD NAME=(X4,X5,X6),ACCESS=INDEX
DATASET DD1=X4P,OVFLW=X40
SEGM NAME=X4A,BYTES=6,PARENT=0
FIELD NAME=(X4F1,SEQ,U),START=1,BYTES=6
LCHILD NAME=(X5FLW,SEQ,U),INDEX=DAF1X
SEGM NAME=X5A,BYTES=6,PARENT=0
FIELD NAME=(X5FLX,SEQ,M),START=1,BYTES=6
LCHILD NAME=(X6FLW,SEQ,M),INDEX=DCF1X
SEGM NAME=X6A,BYTES=6,PARENT=0
FIELD NAME=(X6FLX,SEQ,M),START=1,BYTES=6
LCHILD NAME=(X7FLW,SEQ,M),INDEX=DEF1X
DBDGEN
FINISH
END
```

*Figure 26. Indexed Database, Primary Index Database, and Shared Secondary Index Database DBD Generations*
Chapter 2. Program Specification Block (PSB) Generation

Before executing an application program under IMS, you must describe that program and its use of logical terminals and logical data structures through a program specification block (PSB) generation. 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 used by the application program. In addition, there must be a statement supplying characteristics of the application program itself. There must be one PSB for each message, 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 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.

Related Reading: For more information about GPSBs, see [IMS Version 8: Installation Volume 2: System Definition and Tailoring] and [IMS Version 8: Administration Guide: Transaction Manager].

In this Chapter:

- "Input and Output of PSB Generation"
- "PSBGEN Procedure" on page 113
- "Utility Control Statements for PSB Generation" on page 115
- "Output Messages and Statistics for PSB Generation" on page 136
- "Examples of PSB Generation" on page 137

Input and Output of PSB Generation

PSB generation 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. Before online execution, ACBGEN must be performed to prebuild the expanded PSBs into ACBLIB. PSBLIB is used as input to the ACBGEN process. Batch executions can also use prebuilt blocks from 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 in conjunction 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.
- SENSESEG 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.
PSBGEN

- 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 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.

Related Reading: For more information about PCBs, see IMS Version 8: Application Programming: Database Manager.

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.

**Related Reading:** For more information about PCBs, [IMS Version 8: Application Programming: Database Manager](#).

---

**PSBGEN Procedure**

IMS system definition places the procedure named PSBGEN in the IMS.PROCLIB procedure library.

This two-step assemble and link-edit procedure produces PSBs. The first step, Step C, an operating system assembly, is performed after the procedure is invoked. The second step, Step L, is a link-edit which takes the assembly output from Step C and places the PSBs in IMS.PSBLIB.

**PROC Statement**

The procedure statement is shown in Figure 27. The list following the figure defines the parameters used in the statement.

```plaintext
// PROC MBR=TEMPNAME,SOUT=A,RGN=4M,SYS2=
// EXEC PGM=ASMA90,REGION=&RGN,PARM='OBJECT,NODECK'
// SYSLIB DD DSN=IMS.&SYS2,SDFSMA,C,DISP=SHR
// SYSLIN DD UNIT=SYSDA,DISP=(,PASS),
// SPACE=(80,(100,100),RLSE),
// DCB=(BLKSIZE=80,RECFM=F,LRECL=80)
// SYSPRINT DD SYSOUT=&SOUT,DISP=(,DELETE),
// SPACE=(CYL,(10,5))
// EXEC PGM=IEWL,PARM='XREF,LST',COND=(0,LT,C),REGION=120K
// SYSLIN DD DSN=*.C.SYSLIN,DISP=(OLD,DELETE)
// SYSPRINT DD SYSPRINT DSN=.C.SYSLIN,DISP=(,DELETE),
// SPACE=(121,(300,300),RLSE,ROUND)
// SYSLMOD DD DSN=IMS.&SYS2 PSBLIB(&MBR),DISP=SHR
// SYSLUT DD UNIT=(SYSDA,SEP=(SYSLMOD,SYSLIN)),
// SPACE=(1024,(100,10),RLSE),DISP=(,DELETE)
```

*Figure 27. PSBGEN Procedure Statement*

**MBR=**

Is the name of the PSB generated. This name should be the same as the name specified on the PSBNAME= operand of the PSBGEN statement. If this precaution is not followed, a user ABEND 929 can occur during execution, or message DFS929I (“BLDL FAILED FOR MEMBER”) can be received during an ACBGEN “BUILD PSB” operation.

**SOUT=**

Specifies the SYSOUT class. The default is A.
PSBGEN Procedure

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 operand must be enclosed in quotes and must include a trailing period, for example, SYS2='IMSA.'.

Step C

Step C is the assembly step.

Related Reading: For more information about assembly steps, see High Level Assembler Programmer’s Guide.

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 link-edit 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 link-edit 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 link-edit step with RMODE=24.

Related Reading: For more information about linkage editors, see MVS/DFP Linkage Editor and Loader.

DD Statements

SYSLMOD DD
  Defines an output partitioned data set, IMS.PSBLIB, for the linkage editor.

Invoking the Procedure

The JCL statements in Figure 28 on page 115 are used to invoke the PSBGEN procedure.
Utility Control Statements

Utility Control Statements for PSB Generation

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.
Utility Control Statements

Notes:

1. *label* and PCBNAME are mutually exclusive. Use only the label or the PCBNAME parameter.

For details on the coding format for Assembler macro instructions, refer to the “Assembler Coding Conventions” section in the *IBM Assembler Manual*, publication number SC26-4940-03.

*label*

Specifies an alphanumeric label from 1 to 8 characters long, that is valid for an assembler language PCB 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 keyword for the output message destination. The “name” is the actual destination of the message and is either a logical terminal name (LTERM=) or a transaction-code name (NAME=). When the name is a transaction-code name, output messages to this PCB are enqueued for input to the program used to process the transaction code named by the NAME operand. The name must be from 1- to 8-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= operand is required except when MODIFY=YES is specified.

*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= operand.
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 cancelled.
  
  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.

### DL/I 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. 2500 database PCB definitions are
impractical because this many definitions can require more storage than is usually available. This is the maximum value for application programs executing in all IMS region types (MSG, DL/I, and so on).

The following diagram shows the format for the DL/I database PCB statement.

```
PCB

(1)

label

TYPE=DB, DBDNAME=, NAME=

PROCOPT=A, PROCOPT=

NO, SB=COND

KEYLEN=value

PROCSEQ=index dbname, VIEW=MSDB

LIST=YES, LIST=NO

a:

A

E P H

G

E S P Q N T H

I

E P H

R

E H

D

E P H

L

S R

b:

POS=SINGLE, POS=MULTIPLE
```

Notes:

1. `label` and PCBNAME are mutually exclusive. Use only the label or the PCBNAME= parameter.

2. 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 MVS 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.

**TYPE=DB**

Is a required keyword parameter for all DL/I database PCBs.

**DBDNAME= or NAME=**

Is the parameter keyword 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 PCB statement and precede the next PCB or PSBGEN statement must refer to segments defined in the DBD named in the DBDNAME= or NAME= operand of this PCB. (Refer to "SENSEG Statement" on page 129 for more information.)

The keywords DBDNAME and NAME are synonymous. DBDNAME is more descriptive, and NAME is kept for compatibility with earlier releases.

**PCBNAME=**

Specifies the name of the PCB. The PCB name must be an alphanumeric, 8-byte character string that follows standard naming conventions.

**Exception:** Do not specify this parameter if the PCB statement includes label.

**PROCOPT= (with full function)**

Is the parameter keyword for the processing options on sensitive segments declared in this PCB that you can use in an associated application program. You can use a maximum of four options with this operand. The letters in the operand have the following meaning:

- **A** All, includes the G, I, R, and D functions. PROCOPT=A is the default setting.
- **G** Get function.
- **I** Insert function.
- **R** Replace function. Includes G.
- **D** Delete function. Includes G.
- **P** Position function. 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 required if command code D is used when processing DEDBs. Refer to the section, "Use of PROCOPT=(with Fast Path)" on page 121 for information on how to use PROCOPT=P with DEDBs. P is used in conjunction with A, G, I, D, and L.
- **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 G0, GON, GONP, GOT, GOTP, or GOP only.

**Related Reading:** For more information about the O option, see IMS Version 8: Application Programming: Design Guide.

**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** Is the same as the N operand, 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.

**L** Load function 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** Segments loaded 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 using a particular PSB. The restrictions for using PROCOPT=H are:
- 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 using HSSP uses multiple PCBs with PROCOPT=H for the same database within the same PSB, all database calls using a PCB other than the first one used receive an FH status code. You can use the NOPROCH keyword on the SETO statement to alleviate this restriction.

H is used in conjunction with A, G, I, R, and D.

If you do not specify the PROCOPT operand, it defaults to PROCOPT=A. The replace and delete functions also imply the Get function.
A user abend (U8XX) from the retrieve module (DFSDCR00) 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= operand with an N or a T.

Notes:

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, 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, it is recommended that the data set be 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= operand governs the logical child segment of the concatenated segment. The logical parent of the concatenated segment is governed by the RULES= operand 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.

Use of PROCOPT=(with Fast Path)

In a non-terminal-related or fixed terminal-related MSDB, only the processing options G and R are valid.
Utility Control Statements

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.

Related Reading: For more information about the P processing option or the UOW for DEDBs, see IMS Version 8: Administration Guide: Database Manager.

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 G0, GON, or GOT, and can not be used in conjunction with H.

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 EPSTGOFB.
2. Region 1 issues a call to update the segment. Region 1 PCBA steals the buffer off its EPSTGOFB. 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 EPSTGOFB.
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 EPSTGOFB.
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 EPSTGOFB.
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 EPSTGOFB.
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 EPSTGOFB and the old buffer is still linked to EPSTGOFB. 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
EPSTGOFBF, 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 EPSTGOFBF.
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 EPSTGOFBF.
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
EPSTGOFBF 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=
operand 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.

Related Reading: For more information on the P processing option or the UOW for
DEDBs, see [IMS Version 8: Administration Guide: Database Manager]
If you use the D command code in a call to a DE, the P processing option need not be specified in the PCB for the program.

Procopt H can 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.

**SB=**

Specifies which PCBs will be buffered using sequential buffering (SB). This is an optional keyword. The default is SB=NO, unless the default option has been modified for Batch and BMPs by the DFSSBUX0 to SB=COND.

**Related Reading:** For more information about DFSSBUX0, see [IMS Version 8: Customization Guide](#).

**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.

**Recommendation:** 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. [Figure 29 on page 126](#) 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. Table 10 shows how the KEYLEN= will be specified.

**Table 10. How A KEYLEN Is Determined**

<table>
<thead>
<tr>
<th>Database Hierarchical Paths</th>
<th>Concatenated Key Length Paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>A+B+C=</td>
<td>30 bytes</td>
</tr>
<tr>
<td>A+B+D=</td>
<td>30 bytes</td>
</tr>
<tr>
<td>A+E=</td>
<td>260 bytes</td>
</tr>
<tr>
<td>A+F+G+H+J=</td>
<td>120 bytes</td>
</tr>
<tr>
<td>A KEYLEN=260 bytes would be specified</td>
<td></td>
</tr>
</tbody>
</table>
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=MULTIPLE 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 operand through a secondary processing sequence. The operand is optional. It is valid only if a secondary index exists for this database. If this operand 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 operand.

`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 are 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.

**VIEW=MSDB**

Is used to specify the MSDB commit view. Your existing applications can use either MSDB commit view or the default DEDB commit view. To use the MSDB commit view for DEDBs, specify `VIEW=MSDB` on the PCB statement. If you do not specify `VIEW=MSDB`, the DEDB will use the DEDB commit view. No changes to any existing application programs are required to migrate your MSDBs to DEDBs.

If you issue a REPL call with a PCB that specifies `VIEW=MSDB`, the segment must have a key. This includes any segment in a path if command code 'D' is specified. Otherwise, status AM is returned.

For more information on the `VIEW=MSDB` operand see **IMS Version 8:** Administration Guide: Database Manager.

**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.

See page 117 for information about naming PCBs on the DL/I database PCB statement.
Utility Control Statements

GSAM PCB Statement

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 operand.

- 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 MVS assembler language PCB statement. The labels for the PCB statements within a PSB must be unique.

Exception: Do not specify this parameter if PCBNAME= 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.

**SENSEG Statement**

You use the SENSEG statement with the database 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 statements can be included; each statement must immediately follow the PCB statement to which it is related. There must be one SENSEG statement for each segment to which the application program is sensitive. 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 PCB 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.

The format of the SENSEG statement is as follows:

```
SENSEG NAME=name, PARENT=0 name, PROCOPT=(1) G I E R P D A K
```

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.
Utility Control Statements

PARENT=
Is the segment type name of this segment’s parent.

Requirement: This operand 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 operand 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 operand has the same meaning as the PROCOPT= operand on the statement. In addition to the valid options for this operand listed in [118], 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= operand is not specified, the PCB PROCOPT operand is used as default. If there is a difference in the processing options specified on the PCB and SENSEG statements and the options are compatible, SENSEG PROCOPT overrides the PCB PROCOPT. If PROCOPT= L or LS is specified on the preceding PCB statement, this operand must be omitted.

Do not specify a SENSEG statement for a virtual logical child segment type if PROCOPT= L or LS is specified. The Replace 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 statement.

In the case of concatenated segments, the PROCOPT= operand governs the logical child segment of the concatenated segment. The logical parent of the concatenated segment is governed by the RULES= operand of the SEGM 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 operand) must be 1 through 8. The sensitivity for the pointer (the second operand) must be R (read sensitive) or U (update). If the first operand and the second operand 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= operand can be specified
for indexed segment types only. It enables SSAs of calls for the indexed
segment type to be qualified on the search field of the index segment type
contained in each secondary index specified.

**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= operand of the SENSEG statement for the indexed
segment type or in the PROCSEQ= operand of the PCB statement.

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.

**Figure 30** shows the data structure of segment definition and includes segments A-
F.

![Diagram](image_url)

**Figure 30. Data Structure of Segment Definition**

All of these segments are defined within one DBD. Do not specify INDICES= on a
SENSEG 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:

<table>
<thead>
<tr>
<th>Col. 10</th>
<th>Col. 16</th>
<th>Col. 72</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCB</td>
<td>TYPE=DB, DBDNAME=DATABASE,          X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PROCOPT=A, KEYLEN=22</td>
<td></td>
</tr>
<tr>
<td>SENSEG</td>
<td>NAME=A, PARENT=0, PROCOPT=G</td>
<td></td>
</tr>
<tr>
<td>SENSEG</td>
<td>NAME=B, PARENT=A, PROCOPT=G</td>
<td></td>
</tr>
<tr>
<td>SENSEG</td>
<td>NAME=C, PARENT=B, PROCOPT=I</td>
<td></td>
</tr>
<tr>
<td>SENSEG</td>
<td>NAME=D, PARENT=A, PROCOPT=A</td>
<td></td>
</tr>
<tr>
<td>SENSEG</td>
<td>NAME=E, PARENT=D, PROCOPT=G</td>
<td></td>
</tr>
<tr>
<td>SENSEG</td>
<td>NAME=F, PARENT=D, PROCOPT=A</td>
<td></td>
</tr>
</tbody>
</table>

**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, via 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:

```
SENFLD NAME=name,START=startpos
```

A:

```
REPLACE=YES
```

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 32767.

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.

**PSBGEN Statement**

The PSBGEN statement specifies characteristics of the application program. The following syntax diagram shows the format for the PSBGEN statement.

```
PSBGEN PSBNAME=name,LANG=COBOL,MAXQ=nr
```

Utilities Reference: System
PSBNAME=
Specifies the parameter keyword 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 ACBGEN time.

LANG=
An optional keyword that indicates the compiler language in which the message processing or batch processing program is written. The value for this parameter must be COBOL, PL/I, ASSEM, PASCAL, JAVA, or blank. Leave the value blank if the application has been enabled for the IBM Language Environment® for MVS & VM. If you specify OLIC=YES, LANG=PL/I is invalid. If your application program is written in C language, specify LANG=ASSEM.

CICS and the IBM Language Environment for MVS & VM do not support PASCAL.

You must specify LANG=JAVA for any PSBs associated with a JMP type IMS Java application.

If you are using IMS PL/I applications that run in a compatibility mode using the PLICALLA entry point, you must specify LANG=PLI on the PSBGEN. If you change the entry point and add SYSTEM(IMS) to the EXEC PARM of the compile step, you can specify LANG=blank or LANG=PLI on the PSBGEN. Table 11 shows when to use LANG=blank and LANG=PLI.

Table 11. Using LANG= Option in an LE/370 Environment for PL/I Compatibility

<table>
<thead>
<tr>
<th>Compile exec statement is</th>
<th>and entry point name is PLICALLA</th>
<th>Then LANG= is as stated as follows:</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARM=(...SYSTEM(IMS)...</td>
<td>Yes</td>
<td>LANG=PLI</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>LANG=blank or LANG=PLI</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>Note: Not valid for IMS PL/I applications</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>LANG=PLI</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>LANG=PLI</td>
</tr>
</tbody>
</table>

PLICALLA is only valid for PL/I compatibility support in an LE/370 environment.

If a PL/I application using PLICALLA entry at link-edit time is link-edited using LE/370 with the PLICALLA entry, the link-edit will work; however, you must use
Utility Control Statements

**LANG=PLI.** If the application is re-compiled using PL/I MVS & VM Version 1 Release 1, and link-edited using LE/370 Version 1 Release 2, the link-edit will fail. You must remove the PLICALLA entry statement from the link-edit.

**MAXQ=**
Is the maximum number of database calls with Qx command codes that can be issued between synchronization points. If this number is exceeded, the application program will abend. The default value is zero.

**CMPAT=**
Provides compatibility between BMP or MSG and Batch-DL/I parameter lists. If CMPAT=YES, the PSB is always treated as if there were an I/O PCB, no matter how it is used. If CMPAT=NO, the PSB has an I/O PCB added only for BMP or MSG regions. The default is NO.

**IOASIZE=**
 Specifies the size 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’s copy of the user’s I/O area data during scheduling of this application program. If you do not specify this value, the ACB utility program calculates a maximum I/O area size and uses it as a default. The size calculated is the total length of all 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 OASIZE calculated by the ACBGEN utility. The value of the IOASIZE that will be used is indicated in message DFS0593I issued by ACBGEN. The major components of this pool requirement are IOASIZE and SSASIZE. When the PSB is built into ACBLIB, ACBGEN message DFS0589I indicates the PSB’s total work pool space requirement.

If STAT calls or the test program (DFSDDLTO) is 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 were checkpointed (if this PSB contains any).
- Largest LRECL used in a GSAM data set that is checkpointed.

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, ACBGEN message DFS0589I indicates the PSB’s total work pool space requirement.

**IOEROPN=**

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 you specify the WTOR subparameter, a WTOR for the DFS0451A I/O error message is issued, and DL/I waits for the operator to respond before continuing. If you respond ABEND, IMS terminates with a U451 abend. If you respond CONT IMS continues. Any other response causes the DFS0451A message to be reissued.

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.

If you code the WTOR subparameter, you must code the \( n \) subparameter and parentheses are required. If you code only IOEROPN=\( n \), parentheses are not required.

**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 operand is invalid if any DBPCB (TYPE=DB) specifies PROCOPT=L or LS.

**Exception:** This operand is not applicable to CICS/MVS™ 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 GSAM PCB.
- DB2 reports a deadlock either on a thread create or on an SQL call.
Utility Control Statements

YES means that the internal ROLB call should be done and that the SQL code regarding the deadlock should be returned to the application program. NO means that the internal ROLB call should not be not done and that a user abend 777 should occur. If the GSROLBOK parameter is omitted, 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.

There can be several PCB statements for message output and several PCB statements for databases, but only one PSBGEN statement in a PSB generation statement deck. The PSBGEN statement must be the last statement in the deck preceding the END statement.

END Statement

All PSB generation utility control statements must be followed by an END statement.

Requirement: The END statement is required by the macro assembler to indicate the end of the assembly data.

Output Messages and Statistics for PSB Generation

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 each control statement result in diagnostic messages being printed immediately following the image of the last control statement read before the error was discovered. The message 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 link-edit 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 link-edit of the object module, which produces a load module that becomes a member of IMS.PSBLIB.

**PSB Generation Error Conditions**

See [IMS Version 8: Messages and Codes, Volume 1](#) for a complete description of the IMS messages that indicate PSB errors.

---

**Examples of PSB Generation**

This section includes examples of the use of the PSBGEN utility.

**PSB Generation Examples**

This example shows a PSB generation for a message processing program to process the hierarchic data structure shown in Figure 31. The data structure contains segments: PARTMAST, CPWS, POLN, OPERTON, INVSTAT, and OPERSGMT.

![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.

```plaintext
//PSBGEN JOB MSGLEVEL=1
// 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,PROCLOPT=A,KEYLEN=100
SENSEG NAME=PARTMAST,PARENT=0,PROCLOPT=A
SENSEG NAME=CPWS,PARENT=PARTMAST,PROCLOPT=A
SENSEG NAME=POLN,PARENT=PARTMAST,PROCLOPT=A
SENSEG NAME=OPERTON,PARENT=PARTMAST,PROCLOPT=A
SENSEG NAME=INVSTAT,PARENT=OPERTON,PROCLOPT=A
SENSEG NAME=OPERSGMT,PARENT=INVSTAT,PROCLOPT=A
```

---

*Figure 31. Sample Hierarchic Data Structure*

---

Chapter 2. Program Specification Block (PSB) Generation 137
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 MSGLEVEL=1
// 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 MSGLEVEL=1
// EXEC PSBGEN,MBR=APPLPGM3
//C.SYSIN DD *

PCB TYPE=TP,NAMES=OUTPUT1
PCB TYPE=TP,NAMES=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 MSGLEVEL=1
// 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
```

Examples
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 MSGLEVEL=1
// 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

[Figure 32 on page 140] 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.
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.

Figure 32. Sample Field Level Sensitivity PSB Generation

Fast Path PSB Generation Examples
### Example 2

This example shows the statements for DEDB subset pointers.

```plaintext
//PSBGEN  JOB  MSGLEVEL=1
//  EXEC  PSBGEN,MBR=APPLPGM1
//C.SYSIN  DD  *

PCB  TYPE=DB,DBDNAME=MSDBLM01,PROCOPT=R,  NONTERMINAL-RELATED  X
    KEYLEN=4
SENSEG NAME=LDM,PARENT=0
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
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=DDLTM01  END  OF  PSBGEN
END  END  OF  PSB  GEN
/*

Notes:

1.  SSPT=((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 example in [Figure 33 on page 142](#) 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
Examples

divided into payroll and skill) is at the second level. Address, Payroll, Expr, and Educ are on the third level.

```
//PSBGEN  JOB  MSGLEVEL=1
//     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=PGMX
END
/*
```

Figure 33. A PSBGEN Statement Used to Define a DL/I Database Statement (Example 1)

Example 2: The example in Figure 34 on page 143 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.
Example 3: The example in Figure 35 on page 144 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.
Examples

Example 4: The example in Figure 36 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”.

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 5:** The example in Figure 37 shows either a message switching or conversational message program. The JCL is saved as load module DFSSAM01 in the IMS.PSBLIB library.

```
//PSBGEN JOB MSGLEVEL=1
// EXEC PSBGEN,MBR=APPLPGM1
//C.SYSIN DD *
SENSEG NAME=BACKORDR,PARENT=STOKSTAT
PSBGEN LANG=COBOL,PSBNAME=DFSSAM01
END /*
```

*Figure 37. The Data Structure and JCL For a Message Switching or Conversational Message Program*

**Example 6:** The JCL shown in Figure 38 on page 146 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.
Example 7: The example in Figure 39 on page 147 defines the entire logical structure from the DL/I database DI21PART. The JCL is saved as load module DFSSAM03 in the IMS.PSBLIB library.
Example 8: The example in Figure 40 on page 148 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.
Example 9: The example in Figure 41 on page 149 is identical to example 8, except this JCL is saved as load module DFSSAM06 in the IMS>PSBLIB library.
Example 10: The example in Figure 42 on page 150 defines the entire logical structure from the DL/I database DI21PART. The JCL is saved as load module DFSSAM07 in the IMS.PSBLIB library.
Example of a Shared Secondary Index

**Example 11:** The database structure for this example is shown in [Figure 43 on page 151](#). 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 42. The Data Structure and JCL for a Logical Database Defined From DL/I Database DI21PART**
The database structure for index through DA is shown in Figure 44. It contains segments DA, DB, DC, DD, and DE.

Figure 44. The Data Structure and JCL For Index Through Segment DA

The database structure for index through DC is shown in Figure 45 on page 152. It shows segment DC, DA, DD, and DE.
Examples

This database structure can also include, as a substructure, the database structure for index through DA.

The database structure for index through DE is shown in Figure 46 on page 153. It shows segments DE, DC, and DA.
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 MSGLEVEL=1
//  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=PX4
END
/*
```

Figure 46. The Data Structure and JCL For Index Through Segment DE

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 MSGLEVEL=1
//  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=PX4
END
/*
```
Examples
Chapter 3. Application Control Blocks Maintenance Utility

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

Related Reading: For a description of the DBGEN procedure, see IMS Version 8: Utilities Reference: Database and Transaction Manager.

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.

The Application Control Blocks Maintenance utility saves instruction execution and direct-access wait time and improves performance in application scheduling. It provides a facility for pre-building the required application control blocks off-line; hence, when the application is scheduled, its application control blocks can be read in directly, and control can be passed promptly to the application program.

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 Application Control Blocks 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. This 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.

In this Chapter:

- “Restrictions for ACBGEN” on page 156
- “Input and Output for ACBGEN” on page 156
- “Utility Control Statements for ACBGEN” on page 159
- “Error Processing for ACBGEN” on page 163
- “Examples of ACBGEN” on page 163
Restrictions for ACBGEN

You do not need to run ACBGEN 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 ACBGEN 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 ACBGEN to build the control block for the non-GSAM databases.

IMS conforms to MVS 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.

The Application Control Blocks 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.

When specifying BUILD PSB=ALL on a SYSIN control statement, all PSBs must reside in a single PSBLIB. No concatenated PSBLIBs will be acknowledged on the IMS DD statement.

Input and Output for ACBGEN

Figure 47 on page 157 shows the functional relationship of the I/O data sets and their naming requirements. The Application Control Block Maintenance utility receives input from IMSVS.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 IMSVS.ACBLIB data set.
ACBGEN Procedure

The following procedure is created as a part of system generation. It is placed into the IMS.PROCLIB procedure library by stage two of IMS system definition.

Figure 48 shows the procedure for ACBLIB maintenance.

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

Figure 48. ACBLIB Maintenance Procedure

In Figure 48 the high level qualifier of the IMS data sets is IMS. This 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.

Invoking the Procedure

The following is a sample of the JCL statements that can be used to invoke the ACBGEN procedure.
The ACBGEN 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 depends on the size of the blocks to be generated and typically varies from 100 to 150KB. The default is 256KB.

**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 include a trailing period and be enclosed in quotes, for example: SYS2='IMSA.'

**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.

**DD Statements**

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

**DFSRESLB DD**
Points to an authorized library which 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 reside in an authorized library. The JOBLIB or STEPLIB statement need not be authorized for IMS batch.

**SYSPRINT DD**
Defines the output message data set.

**IMS DD**
Defines the IMS.PSBLIB and IMS.DBDLIB data sets.
IMSACB DD
Defines the IMS.ACBLIB data set.

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

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

Related Reading: Refer to the OS/390 V2R10.0 DFSMSdfp™ Utilities, SC26-7343 manual for more information about space allocation requirements.

SYSUT4 DD
Same function as SYSUT3.

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

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.

Recommendation: It is suggested that this data set reference a member of IMS.PROCLIB containing the following required control statement:

```
//COMPCTL DD DSNAME=IMS.PROCLIB(DFSACBCP), DISP=SHR
```

SYSIN DD
Defines the input control statement data sets. They can reside 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.

DFSACBCP Control Statement
The following control statement is created as a part of system generation and is placed in the IMS.PROCLIB procedure library by stage two of IMS system definition.

The ACBGEN procedure uses DFSACBCP to compress ACBLIB.
COPY INDD=IMSACB,OUTDD=IMSACB

Utility Control Statements for ACBGEN
The following guidelines apply to the utility control statements for ACBGEN:

- The utility control statements for this program are coded with one restriction: To continue a statement, enter a non-blank character in column 72 and begin the statement on the next line starting in column 16.
- A statement is coded as a card image and is contained in columns 1 to 71.
- The control statement can optionally contain a name, starting in column 1.
- The operation field must be preceded and followed by one or more blanks.
- The operand is composed of one or more PSB/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 operand of a control statement, separated from the operand by one or more blanks.
Utility Control Statements

ACB Maintenance Utility Syntax: Build format

BUILD
Indicates that blocks are to be built for the named PSBs which refer to the named DBDs.

DELETE
Indicates that blocks are to be deleted from ACBLIB. The named PSBs and all PSBs that refer to the named DBDs are deleted.

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

Requirement: When you specify BUILD PSB=ALL on a SYSIN control statement, all PSBs must reside in a single PSBLIB. No concatenated PSBLIBs will be acknowledged on the IMS DD statement.

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

DBD=(dbdname)
Means blocks are to be built or deleted for this DBD and for all PSBs which reference this DBD either directly or indirectly through logical relationships. The DBD to be built must already exist in IMS.ACBLIB. The referencing PSBs must already exist in IMS.ACBLIB. PSBs newly added to IMS.PSBLIB must be referenced by PSB= operands. Because deleting a PSB does not delete any DBDs referenced by the PSB, this operand can be used to delete specific DBDs. However, deleting or building a DBD causes every PSB in IMS.ACBLIB 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 operand.

Example: 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 ACBGEN, 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 ACBGEN. In summary, even though the referencing PSB is completely updated, but the updated DBDs must be explicitly specified in the ACBGEN.

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 generate a member in IMS.ACBLIB and cannot be referenced on BUILD or DELETE control statements.

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.

Figure 49 on page 162 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.
Managing Dynamic Option (DOPT) PSBs

Using dynamic option (DOPT) PSBs requires concatenation of the following ACBLIB data sets:

- A primary ACBLIB data set to contain blocks for all non-dynamic PSBs
- A DOPT ACBLIB data set to contain blocks for all dynamic option PSBs

The primary ACBLIB data sets 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. Remember, 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.
Error Processing for ACBGEN

The ACBGEN procedure returns the following codes:

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion of all operations</td>
</tr>
<tr>
<td>4</td>
<td>One or more warning messages issued</td>
</tr>
<tr>
<td>8</td>
<td>One or more blocks could not be built</td>
</tr>
<tr>
<td>16</td>
<td>Program terminated due to severe errors</td>
</tr>
</tbody>
</table>

Examples of ACBGEN

Example 1

This example creates blocks for all PSBs that currently reside in IMS.PSBLIB. All blocks currently existing in IMS.ACBLIB are deleted and their space is reused. 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.

```bash
//BLDBLKS JOB 1,1,MSGLEVEL=(1,1)
/*/STEP EXEC ACBGEN,SOUT=A
//SYSIN DD *
   BUILD PSB=ALL
/*
```

Example 2

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.

```bash
//BLDBLKS JOB 1,1,MSGLEVEL=(1,1)
/*/STEP EXEC ACBGEN,SOUT=A,COMP=POSTCOMP
//SYSIN DD *
   BUILD PSB=(PSB1,PSB2,PSB3)
   DELETE DBD=(DBD5,DBD6)
/*
```

Example 3

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 1,1,MSGLEVEL=(1,1)
//*
//STEP EXEC ACBGEN,SOUT=A,COMP='PRECOMP,POSTCOMP'
//SYSIN DD *
  DELETE PSB=PSB1
  BUILD DBD=DBD4
/*
Part 2. Service Utilities

Chapter 4. Dynamic Allocation Macro (DFSMDA) ............ 167
Restrictions for DFSMDA .................................. 169
Input and Output for DFSMDA ............................. 169
IMSDALOC Procedure
  PROC Statement ............................................. 170
  JCL Parameter Description ................................. 170
  Step ASSEM ................................................... 171
  Step BLDMBR ............................................... 171
  Step LNKEDT ............................................... 171
  Invoking the Procedure .................................... 171
Macro Statements for DFSMDA .............................. 172
Examples of DFSMDA
  Example 1 .................................................... 177
  Example 2 .................................................... 177
  Example 3 .................................................... 178
  Example 4 .................................................... 178
  Example 5 .................................................... 178

Chapter 5. Security Maintenance Utility (DFSISMPO) ........... 179
Input and Output Flow for DFSISMPO ....................... 180
Restrictions for DFSISMPO ................................ 181
Security Options for DFSISMPO ............................ 181
  LTERM Security ............................................. 182
  Password Security ......................................... 182
  Transaction Command Security ............................ 182
  IMS Resource Access Security ............................ 182
  Sign-on Verification Security ............................ 183
  Resource Access Control Facility (RACF) ................ 183
  Command Authorization Exit Routine .................... 183
  IMS Application Resource Access Security ............. 183
SECURITY Procedure
  PROC Statement ............................................. 184
  JCL Parameter Description ................................. 186
  Step S EXEC Statement ..................................... 186
  DD Statements .............................................. 187
  Step C ....................................................... 187
  Step L ....................................................... 188
  Invoking the Procedure .................................... 188
Utility Control Statements for DFSISMPO ................ 188
Output for DFSISMPO ...................................... 190
  Security-Status Reports ................................. 191
Examples of DFSISMPO ..................................... 191
  Example 1 .................................................... 191
  Example 2 .................................................... 191
  Example 3 .................................................... 192
  Example 4 .................................................... 192
  Example 5 .................................................... 193
  Example 6 .................................................... 194
  Example 7 .................................................... 194
  Example 8 .................................................... 194

Chapter 6. Online Change Utilities and Procedures .......... 197
Online Change Copy Utility (DFSUCU0) ..................... 197

© Copyright IBM Corp. 1974, 2005
<table>
<thead>
<tr>
<th>Chapter 7. Dynamic SVC Utility (DFSUSVC0)</th>
<th>209</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restrictions for DFSUSVC0</td>
<td>209</td>
</tr>
<tr>
<td>Input and Output for DFSUSVC0</td>
<td>209</td>
</tr>
<tr>
<td>Error Processing for DFSUSVC0</td>
<td>210</td>
</tr>
<tr>
<td>JCL Requirements for DFSUSVC0</td>
<td>210</td>
</tr>
<tr>
<td>DD Statements</td>
<td>210</td>
</tr>
<tr>
<td>Example of DFSUSVC0</td>
<td>211</td>
</tr>
</tbody>
</table>
Chapter 4. Dynamic Allocation Macro (DFSMDA)

Use the Dynamic Allocation macro (DFSMDA) to build a member (that is, one or more parameter lists) for naming data sets that can participate in dynamic allocation and deallocation.

**Related Reading:** Refer to [IMS Version 8: Installation Volume 1: Installation Verification](#) for more information about IMS.SDFSRESL.

IMS users and CICS users can dynamically allocate IMS databases. To use DFSMDA you must catalog all specified database data sets. However, you do not need to initially allocate them through control region JCL.

For Fast Path databases, if the database data sets to be allocated are registered in DBRC, the information required to dynamically allocate the data sets is obtained from DBRC. You do not need to supply DFSMDA members for them. When the dynamic allocation information is obtained from DBRC, the DISP= used to allocate the data sets is either DISP=OLD or DISP=SHARE depending on the following:

- If SHARELVL=0 or RECONS, use DISP=OLD.
- If SHARELVL=1, 2, or 3 or RECONS, use DISP=SHARE.

**Related Reading:** For more information about data sharing levels, see the [CHANGE.DB command in IMS Version 8: Database Recovery Control (DBRC) Guide and Reference](#).

The priority of allocation information is shown in [Table 12](#).

**Table 12. Allocation Information Priorities**

<table>
<thead>
<tr>
<th></th>
<th>DBRC</th>
<th>DD Statement</th>
<th>DFSMDA Member</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEEDBs and MSDBs</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>All others</td>
<td>N/A</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Database data sets specified in DFSMDA are allocated at different times depending on whether you are running in an IMS DB/DC, IMS Batch, CICS/MVS 2.1, or CICS/ESA 3.1 environment. The environment requirements are:

- **IMS DB/DC** database data sets are allocated either when a /START command is issued for the database or when an IMS application program is scheduled. You deallocate the data set by the /DBR command. If a database data set is specified in the JCL, it is allocated by MVS during control region startup. You can deallocate it with the /DBR command and reallocate it with the /START command.

- **IMS Batch** database data sets are allocated near the beginning of the job step, before the batch application starts execution.

Dynamic allocation is always attempted for all non-JCL allocated databases defined in the PSB being executed. This is performed by searching the JOBLIB/STEPLIB concatenation for DFSMDA members, unless dynamic allocation is disabled (for batch only) by the presence of the NODYNALLOC statement in your DFSVSMxx member.

If a batch job uses a PSB with more database PCBs than are necessary for a particular job, you can avoid dynamic allocation of the unnecessary databases while still maintaining a library of DFSMDA members for all databases belonging to the PSB. You have two methods of doing this:
Dynamic Allocation

- You can include the NODYNALLOC statement in your DFSVSMxx member and include DD statements for only the necessary databases in your job JCL. The library of DFSMDA members does not need to be removed from the JOBLIB/STEPLIB concatenation because the NODYNALLOC statement disables batch dynamic allocation.

- You can maintain separate libraries of DFSMDA members, which can be included or excluded from the JOBLIB/STEPLIB concatenation as needed. DFSMDA members need not be kept in your IMSVS.SDFSRESL. For example, you can maintain one main library of DFSMDA members for all the databases for a PSB and maintain several subset libraries. You concatenate only the library that is appropriate for the job being run. Dynamic allocation searches the entire JOBLIB/STEPLIB concatenation for DFSMDA members, so you must remove or alter all libraries that contain undesired members.

If the databases for which your program has update intent have logical relationships or secondary indexes, those additional databases containing the logical relationships or secondary indexes can also be allocated, whether by JCL or DFSMDA members. To cause dynamic allocation of a logically related database, change the PROCOPT to indicate update intent. To dynamically allocate a secondary index, change the PROCOPT to indicate update intent or include a PCB with PROCSEQ= for the secondary index.

If the PCB specifies a PROCOPT that does not indicate update intent, no intent will be propagated to a logically related database or to a secondary index, and dynamic allocation will not be attempted for either of these related databases.

- **CICS** database data sets are allocated when an application program issues a schedule call for the PSB.Deallocation occurs, for example, during the processing of STOP and RECOVERDB commands issued against the database.

You can dynamically allocate online log data sets (OLDS), write ahead data sets (WADS), and system log data sets (SLDS) if they are named in the DFSMDA macro. The DFSMDA macro must be defined to permit SLDS input to IMS to restart in MVS.

When you start an OLDS using the /START command, the OLDS must be defined in the DFSMDA macro, even if it is allocated in JCL.

**Related Reading:** Refer to [IMS Version 8: Command Reference](#) for descriptions of how the data sets specified in the DFSMDA macro are treated by the /START, /STOP, and /DBR commands.

The IMS Monitor data set can also participate in dynamic allocation and deallocation. The IMS Monitor data set is allocated when it is started with the /TRACE ON command and deallocated when it is stopped with the /TRACE OFF command. It need not be initially allocated through JCL. It must not be cataloged if residing on tape; it must be cataloged if on DASD.

**Recommendation:** If you use the multiple DEDB area data set facility, it is recommended that you register all data sets belonging to that area in either DBRC or DFSMDA.

The specified areas are allocated either when a /START command is issued for the area or when an application program attempts to use the area. The area is deallocated by /STOP AREA. Multiple areas can be deallocated by /STOP ADS.
In an XRF environment, all database and area data sets must be dynamically allocated.

In this Chapter:

- "Restrictions for DFSMDA"
- "Input and Output for DFSMDA"
- "IMSDALOC Procedure" on page 170
- "Macro Statements for DFSMDA" on page 172
- "Examples of DFSMDA" on page 177

**Restrictions for DFSMDA**

The following restrictions apply when using the Dynamic Allocation macro:

- HALDBs are dynamically allocated and do not need the Dynamic Allocation Macro.
- If you are going to dynamically allocate a database, all DD statements referenced in the DMB for the database must be defined in the TYPE=DATASET, DDNAME= parameter. A database cannot be partially allocated by JCL and partially allocated by a dynamic allocate member.
- Because dynamic allocation cannot resolve logical relationships between DBDs, you must define a dynamic allocation member for each DBD in a logically related database. For example, a HIDAM database is composed of two logically related DBDs, the index DBD and the data area DBD. Each DBD in this example must have a dynamic allocation member with the same name as the DBD.
- The Batch Backout utility (DFSBBO00) is the only IMS utility that is supported for dynamic allocation.
- A database that is generated as a DFSMDA member cannot be given a name that is a duplicate of any label name that is generated during the assembly step of the DFSMDA job. IMS generates a label using the database name during this step, and an error occurs if that label name already exists in code invoked by DFSMDA. This restriction does not apply to data set names.
- A database that is generated as a DFSMDA member cannot be defined with a DDNAME that is identical to the DDNAME defined for another database during the same assembly step of the DFSMDA job. If more than one database must be defined with the same DDNAME (as in the case of secondary indexes), the DFSMDA job must be run separately for each required occurrence of the DDNAME.

**Input and Output for DFSMDA**

The input to the DFSMDA macro consists of statements as explained in "Macro Statements for DFSMDA" on page 172.

The output from the DFSMDA macro consists of text decks and linkage editor statements that are used to create load modules in IMS.SDFSRESL. Batch load modules must be created within IMS.SDFSRESL. Online load modules can be created either in IMS.SDFSRESL or in an unauthorized library.

The members for dynamic allocation can be changed simply by regenerating parameter lists with new input.
 Unless it is a dynamic allocation member, no member that has the same name as a database should be link-edited into IMS.SDFSRESL.

**IMSDALOC Procedure**

The IMSDALOC procedure is created as a part of system generation and is placed into the IMS.PROCLIB library by stage two of IMS/ESA system generation.

This is a three step procedure for generating the list of databases and DEDB data areas that are to be dynamically allocated.

IMSDALOC assumes:
- Input is read from SYSIN.
- Each database or DEDB data set described in the input has a corresponding module placed in the dynamic allocation member data set.
- The name given to each module is the name of the database or DEDB data area described in the input.

**PROC Statement**

Figure 50 shows the JCL for the IMSDALOC procedure. The parameters are described in "JCL Parameter Description."

```plaintext
// PROC SOUT=A, SYS2=
// ASSEM EXEC PGM=ASMA90, PARM='ALIGN, DECK, NOBJECT'
// SYSLIB DD DSN=IMS.&SYS2.SDFSMAC, DISP=SHR
// DD DSN=SYSI.SDFSMAC, DISP=SHR
// SYSUT1 DD UNIT=SYSDA, SPACE=(CYL, (10, 5))
// SYSPUNCH DD DSN=&OBJJMOD, DCB=(RECFM=FB, LRECL=80, BLKSIZE=400),
// SPACE=(400, (100, 100)), UNIT=SYSDA, DISP=(NEW, PASS)
// SYSPRINT DD SYSOUT=&SOUT
// BLDMBR EXEC PGM=IEBUPDTE, PARM='NEW', COND=(7, LT, ASSEM)
// SYSPRINT DD DUMMY
// SYSUT2 DD DSN=&TEMPPDS, UNIT=SYSDA, DISP=(NEW, PASS, DELETE),
// SPACE=(80, (1000, 500, 10)), DCB=(RECFM=F, BLKSIZE=80)
// SYSIN DD DSN=*.ASSEM.SYSPUNCH, DISP=(OLD, DELETE, DELETE)
// LNKEDT EXEC PGM=IEWIL, PARM='LIST, XREF, LET',
// COND=((7, LT, ASSEM), (3, LT, BLDMBR))
// SYSUT1 DD UNIT=SYSDA, SPACE=(1024, (100, 50))
// SYSLIB DD DUMMY
// SYSPRINT DD SYSOUT=&SOUT
// SYSLMOD DD DSN=IMS.&SYS2.SDFSRESL, DISP=SHR
// OBJJMOD DD DSN=&TEMPPDS, DISP=(OLD, DELETE, DELETE)
// SYSLIN DD DSN=&TEMPPDS(LNKCTL), DISP=(OLD, DELETE, DELETE),
// VOL=REF=*.OBJMOD

Figure 50. JCL for the IMSDALOC Procedure

**Recommendation:** The SPACE parameter should be increased to accommodate large volumes of TYPE=DATABASE statements.

**JCL Parameter Description**

**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 operand must be enclosed in quotes and must include a trailing period; for example, SYS2='IMS.A.'.

Step ASSEM
Step ASSEM is the assembly step.

**Related Reading:** Refer to *High Level Assembler Programmer’s Guide* for information on assembly steps.

**DD Statements**

**SYSIN DD**
Defines the input data sets to step C. These DD statements must be provided when invoking the procedure.

Step BLDMBR

**Related Reading:** Refer to the IEBUPDTE utility in *MVS/DFP Utilities* for information on this step.

Step LNKEDT
Step LNKEDT is the link-edit step.

**Related Reading:** Refer to *MVS/DFP Linkage Editor and Loader* for information on linkage-editors.

**DD Statements**

**SYSLMOD DD**
Defines an output partitioned data set for the linkage editor.

For batch execution, the data set must be concatenated with IMS.SDFSRESL, and can be either an authorized library, or an unauthorized data set. To use an authorized library, you must include the DFSRESLB DD statement in the batch execution procedure.

For online execution, or in a DBCTL environment, the data set can be an authorized data set included in the IMS.SDFSRESL concatenation, or an unauthorized data set. To use an unauthorized data set, you must define it to the control region by adding an IMSDALIB DD statement to the online or DBCTL execution procedure. This unauthorized data set will then take precedence over the IMS.SDFSRESL concatenation when seeking a dynamic allocation parameter list or member.

**Invoking the Procedure**
The dynamic allocation macro statements are supplied as input to the IMSDALOC procedure and executed as an MVS job.

**Requirement:** A JOB statement (defined by the using installation), an EXEC statement, and DD statements that define the input and output data sets are required.

The following JCL statement invokes the IMSDALOC procedure.

```
//DALOC    JOB
//*         
//STEP     EXEC IMSDALOC
//*         
```
IMSDALOC Procedure

//SYSIN DD *
 DFSMDA TYPE=
END
 /*

EXEC
 Should be in this form:
 //STEP EXEC IMSDALOC

SYSIN DD
 Defines the input data set containing the DFSMDA macro statements.

Macro Statements for DFSMDA

The DFSMDA macro is coded as an MVS macro. The statement label is optional, the macro "DFSMDA" is coded after one or more blanks, and additional parameters are separated by blanks. MVS continuation rules apply.

The DFSMDA macro has several statement types (as indicated by the TYPE= parameter), each of which uses different additional parameters. Code the statements types as follows:
1. One TYPE=INITIAL statement to start the parameter list build
2. As many TYPE=DATABASE, TYPE=DATASET, and TYPE=FPDEDB statements as necessary
3. One TYPE=DFSDCM_MON if the IMS Monitor data set is to be included
4. One TYPE=FINAL to end the list

The maximum number of TYPE=DATABASE statements allowed is 250. Explanations of all the DFSMDA statement types follow:

**TYPE=INITIAL Statement**
This statement indicates the start of a parameter list build and is required. No other parameters are valid on a TYPE=INITIAL statement. The format of this statement is:

```
 DFSMDA TYPE=INITIAL
```

**TYPE=DATABASE Statement**
This statement specifies the start of the definition for a database to participate in dynamic allocation and deallocation: one or more TYPE=DATASET statements should follow. (Do not use this statement for a DEDB area.) The format of the statement is:

```
 DFSMDA TYPE=DATABASE, DBNAME=dbname
```

**DBNAME=**
Specifies the DBD name of a database whose data sets are to be dynamically allocated. This name is used as a member name in IMS.SDFSRESL to identify this database parameter list. Care should be taken to ensure that this name does not conflict with already existing members in IMS.SDFSRESL. This includes, but is not limited to, IMS modules and user-supplied exit routines.

**TYPE=FPDEDB Statement**
This statement defines an area within a data entry database (DEDB). One TYPE=FPDEDB statement is required for each area to be specified. The format of this statement is:
DBNAME=
Specifies the DBD name of the DEDB in which the specified area resides. This parameter is optional, and is used for documentation purposes only. For DEDB areas, the IMS.SDFSRESL parameter list is not named with the database name, but rather with the area's DDname.

TYPE=DATASET Statement
This statement defines either a data set within the database specified in the previous TYPE=DATABASE statement or a DEDB area. One complete TYPE=DATASET is used for each data set or area data set defined. Every data set within a database to be dynamically allocated and deallocated must be named in a TYPE=DATASET statement. When defining DEDB areas, a TYPE=FPDEDB statement must precede each TYPE=DATASET statement.

If the data set within a database identifies a secondary index data set shared with another database, the DFSMDA members for the two databases must be generated in separate assemblies.

The format of this statement is:

```
DFSMDA TYPE=DATASET, DSNAME=dsname, DDNAME=ddname, DISP=OLD, DISP=SHR
```

DSNAME=
Specifies the name of the data set. The name can be any combination of simple and compound names valid in JCL, except the name cannot contain special characters.

DDNAME=
Specifies the name of the DD statement defining this data set. This name is the same as that used in the DATASET or AREA statement of the DBDGEN.

For multiple ADSs, this name is the same as the ADDN name registered in the ADS RECON data set.

DISP=
Specifies the disposition of this data set when allocated. The default is OLD.

TYPE=DFSDCMON Statement
This statement defines the dynamic allocation parameter list for the IMS Monitor data set. The format of this statement is:

```
DFSMDA TYPE=DFSDCMON, DSNAME=dsname, DDNAME=ddname, DISP=OLD, DISP=SHR
```

A:
```
A: 27992 32768
```
Macro Statements

**DSNAME=**
Specifies the name of the data set, which must not be cataloged if the unit defines a TAPE device. However, if UNIT=DASD is specified, then the data set must be cataloged and available. The name can be any combination of simple and compound names valid in JCL, but must not contain special characters.

**DDNAME=**
Is the required value for DDNAME.

**UNIT=**
Specifies the unit for the DC Monitor data set. If the data set resides on a direct access device, UNIT=DASD must be specified and the data set must be cataloged. Otherwise, the value of UNIT= can be the name of any tape device valid to the installation. The default is UNIT=TAPE.

**BUFNO=**
Specifies the number of buffers for the IMS Monitor data set. Valid numbers range from 2 to a maximum of 99 for DFP and 255 for DFSMS. The default is 4.

**BLKSIZE=**
Specifies the block size for the IMS Monitor data set. For a tape device (UNIT≠DASD), the default is 32K. If UNIT=DASD, the default is 28,332.

**DISP=**
Specifies the disposition for the IMS Monitor data set for a UNIT=DASD data set definition. Valid values are OLD and SHR with OLD being the default if this parameter is not supplied. A warning message is issued if any other value is supplied, and a DISP=OLD value will override the value specified.

**TYPE=RECON Statement**
This statement defines the dynamic allocation parameter list for database recovery control (DBRC).

The format of this statement is:

```
 DFSMDA TYPE=RECON, DSNAME=dsname, DDNAME=RECONn, WAIT=NO, YES
```

**DSNAME=**
Specifies the name of the data set. The name can be any combination of simple and compound names valid in JCL, except that it cannot contain special characters.

**DDNAME=**
Specifies the name of the DD statement defining this data set. This name must be RECON1, RECON2, or RECON3.

**WAIT=**
If YES is specified on any of the TYPE=RECON statements (RECON1, RECON2, RECON3), a wait is issued for any of the RECONs found to be offline during DBRC initialization. WAIT=NO is the default. Omitting the WAIT= parameter or specifying WAIT=NO causes dynamic allocation to fail in the event that a RECON data set is offline during DBRC initialization.
**TYPE=OLDS Statement**

This statement defines the dynamic allocation parameter list for the online log data set (OLDS).

There must be as many DFSMDA macros as there are OLDS.

**Requirement:** If you use dual logging, DFSMDA member names are required for both the primary and secondary OLDS.

The format of this statement is:

```
---DFSMDA---TYPE=OLDS,DSNAME=dsname,DDNAME=DFSOxnn---
```

**DSNAME=**

Specifies the name of the data set. The name can be any combination of simple and compound names valid in JCL, except that it cannot contain special characters.

**DDNAME=**

Specifies the OLDSs to be allocated. If the OLDSs are dual, there must be a pair of macros, one with the DDname of the primary OLDS and the other with the DDname of the secondary OLDS (for example, DFSOLP01 and DFSOLS01). The data set must be cataloged. Substitute P for x when declaring a primary data set. Substitute S for x when declaring a secondary data set. Values from 00 through 99 can be specified for nn.

**TYPE=SLDS Statement**

This statement defines the dynamic allocation parameter list for the SLDS. SLDSs are dynamically allocated when required as input for restart. A single DFSMDA member with name IMSLOGR must be created to specify the UNIT information required for allocation. All SLDSs to be used as input to restart must reside on the same device type.

The format of this statement is:

```
---DFSMDA---TYPE=SLDS,UNIT=device type,DDNAME=IMSLOGR---
```

**UNIT=**

Specifies the device required for allocation. All SLDSs used as input for restart must reside on the same device type. This applies to both the primary and secondary data sets when dual logging is used. The device type can be tape or DASD.

**DDNAME=IMSLOGR**

Is the required value for DDNAME.

**TYPE=TRACE Statement**

This statement defines the dynamic allocation parameter for external trace data sets. External trace data can be written to disk storage or to a tape unit. The statement differs with the type of storage chosen.

The format of the statement for disk allocation is:

```
---DFSMDA---TYPE=TRACE,DDNAME=DFSTRA0n,DSNAME=dsname---
```

**DDNAME**

Specifies the DDname of the data set located on the disk. n specifies the number of the data set, and must be either 1 or 2. Each data set must be cataloged. Use two data sets to ensure that trace data is available at EOV.
Macro Statements

**DSNAME**

Specifies the data set name. The name can be any combination of alphanumeric characters that is valid for IMS, except for special characters such as @, $, or #. DSNAME can be up to 44 bytes long.

The format for data set allocation to tape is:

```
/SM590000/SM590000
 DFSMDA

 TYPE=TRACE,DDNAME=DFSTRA0T,DSNAME=dsname,UNIT=nnnn

 BLKSIZE=nnnnn

/SM590000/SM630000
```

**DDNAME**

Specifies the DDname of the data set located on the tape. DFSTRA0T is the required DDname if you allocate the external trace data set to a tape.

**DSNAME**

Specifies the data set name. DSNAME must not be cataloged. The name and can be any combination of alphanumeric characters that is valid for IMS, except for special characters such as @, $, or #. The name you specify can be up to 44 bytes long.

**UNIT**

Specifies the unit for the external trace data set. The unit must be a tape device, but can be any name valid to the installation. The default value is TAPE.

**BLKSIZE**

Is the block size of the external trace data set. The minimum value is 4008. Any other value chosen must be a multiple of 4004 (the LRECL) plus 4. The default is 16384.

**Recommendation:** IBM now recommends 20024. The recommended BLKSIZE for current DASD is 20024 because it is 1/2 track.

Future DASD might change the track size, and older DASD might have different track sizes.

**TYPE=WADS Statement**

This statement defines the dynamic allocation parameter list for the write ahead data set (WADS).

There must be as many DFSMDA macros as there are WADS data sets.

**Requirement:** If dual logging is used, DFSMDA member names are required for both the primary and secondary WADS.

The DFSMDA member name must be the same as the DDname of the WADS that it defines.

The format of this statement is:

```
/SM590000/SM590000
 DFSMDA

 TYPE=WADS,DSNAME=dsname,DDNAME=DFSWADS

/SM590000/SM630000
```

**DSNAME**

Specifies the name of the data set. The name can be any combination of simple and compound names valid in JCL, except that it cannot contain special characters. The data set must be cataloged.
DDNAME=

Specifies the WADS to be allocated. Values 0 through 9 can be specified for n. When dual logging for the WADS is requested via the WADS=D execution time parameter, there must be at least two WADS provided.

TYPE=FINAL Statement

This statement indicates the end of a parameter list build and is required. No other parameters are valid on a TYPE=FINAL statement. The format of this statement is:

```
<<DFSMDA--TYPE=FINAL-->>
```

Examples of DFSMDA

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

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

Example 1

Example 1 shows the JCL and macro statements to specify three databases and the IMS Monitor data set to participate in dynamic allocation and deallocation.

```
//DALOC  JOB
/*
//STEP   EXEC IMSDALOC
/* +----1----+----2----+----3----+----4----+----5----+----6----+----7---*
//SYSIN  DD *
DFSMDA TYPE=INITIAL
DFSMDA TYPE=DATABASE,DBNAME=DI41M101
DFSMDA TYPE=DATASET,DSNAME=IMSQA.M1I31I,DDNAME=M1I3I1
DFSMDA TYPE=DATASET,DSNAME=IMSQA.M1I30I,DDNAME=M1I3O1
DFSMDA TYPE=DATABASE,DBNAME=DX41SK03
DFSMDA TYPE=DATASET,DSNAME=IMSQA.DB5H111,DDNAME=DXSK0301, DISP=SHR
DFSMDA TYPE=DATASET,DSNAME=IMSQA.DB5H222,DDNAME=DXSK0302, DISP=SHR
DFSMDA TYPE=DATASET,DSNAME=IMSQA.D5H233,DDNAME=DHSK0301, DISP=SHR
DFSMDA TYPE=DATABASE,DBNAME=DH41SK03
DFSMDA TYPE=DATASET,DSNAME=IMSQA.DB4D111,DDNAME=DSDK0101, DISP=SHR
DFSMDA TYPE=DATASET,DSNAME=IMSQA.DB4D222,DDNAME=DSDK0102, DISP=SHR
DFSMDA TYPE=DFSDCMON,DDNAME=IMSMON,DSNAME=I115T237.IMSMON
DFSMDA TYPE=FINAL
END
/*
```

Example 2

Example 2 shows the JCL and macro statements to specify three DEDB areas to participate in dynamic allocation and deallocation.

```
//DALOC  JOB
/*
//STEP   EXEC IMSDALOC
/* +----1----+----2----+----3----+----4----+----5----+----6----+----7---*
//SYSIN  DD *
DFSMDA TYPE=INITIAL
DFSMDA TYPE=FPDEDB,DD41SK02
DFSMDA TYPE=DATASET,DSNAME=DB9AREA0,DDNAME=DB9AREA0
```

Chapter 4. Dynamic Allocation Macro (DFSMDA) 177
Examples

Example 3

In MVS, SLDSs are dynamically allocated when required as input for restart. Example 3 shows the JCL and macro statements for SLDS to participate in dynamic allocation and reallocation.

```jcl
/*
Example 3 shows the JCL and macro statements for SLDS to participate in dynamic allocation and reallocation.

//ASSMBLY EXEC IMSDALOC
//
//SYSLIB DD DSNAME=RNC.SDFSMAC,DISP=SHR
// DD DSNAME=I13OTS13.SDFSMAC,DISP=SHR
// DD DSNAME=SYS1.SDFSMAC,DISP=SHR
//*
//SYSIN DD *
 DFSMDA TYPE=INITIAL
 DFSMDA TYPE=SLDS,UNIT=SYSDA,DDNAME=IMSLOGR
 DFSMDA TYPE=FINAL
 END

//LNKEDT.SYSLMOD DD DSNAME=IMSQA.TNUC2,DISP=SHR,
// UNIT=SYSDA,VOL=SER=USER01
*/
```

Example 4

Example 4 shows the statements required to build allocation members for RECON data sets and to trace data sets on DASD.

```jcl
/*
Example 4 shows the statements required to build allocation members for RECON data sets and to trace data sets on DASD.

//DYNALL JOB
//*
//SYSIN DD *
 DFSMDA TYPE=INITIAL
 DFSMDA TYPE=RECON,DSNAME=IMSV41.RECON01,DDNAME=RECON1,WAIT=YES
 DFSMDA TYPE=RECON,DSNAME=IMSV41.RECON02,DDNAME=RECON2,WAIT=YES
 DFSMDA TYPE=RECON,DSNAME=IMSV41.RECON03,DDNAME=RECON3,WAIT=YES
 DFSMDA TYPE=TRACE,DDNAME=DFSTRA01,DSNAME=IMS41.DFSTRA01
 DFSMDA TYPE=TRACE,DDNAME=DFSTRA02,DSNAME=IMS41.DFSTRA02
 DFSMDA TYPE=FINAL
 END

*/
```

Example 5

Example 5 shows the statements required to build allocation members for RECON data sets and to trace data sets on tape.

```jcl
/*
Example 5 shows the statements required to build allocation members for RECON data sets and to trace data sets on tape.

//DYNALL JOB
//*
//SYSIN DD *
 DFSMDA TYPE=INITIAL
 DFSMDA TYPE=RECON,DSNAME=IMSV41.RECON01,DDNAME=RECON1,WAIT=YES
 DFSMDA TYPE=RECON,DSNAME=IMSV41.RECON02,DDNAME=RECON2,WAIT=YES
 DFSMDA TYPE=RECON,DSNAME=IMSV41.RECON03,DDNAME=RECON3,WAIT=YES
 DFSMDA TYPE=TRACE,DDNAME=DFSTRA0T,DSNAME=TAPEDS1,UNIT=TAPE,BLKSIZE=20024
 DFSMDA TYPE=FINAL
 END

*/
Chapter 5. Security Maintenance Utility (DFSISMP0)

For security beyond that provided by default terminal security, you can use the various security options specified with the Security Maintenance utility (SMU). The utility is executed offline after completion of Stage 2 processing for system definition. Its output is a set of secured-resource tables placed on the IMS.MATRIX data set. The tables are loaded at system initialization time, and, for certain options, work with exit routines and/or the Resource Access Control Facility (RACF®) program product during online execution to provide resource protection.

The Security Maintenance utility provides five security options:

LTERM security
- Defines the commands and transactions that can be used from a given physical or logical terminal

Password security
- Limits the use of a specified IMS resource to someone who supplies the correct password

Resource access security
- Limits the set of IMS resources which can be used by dependent regions that are authorized to access a specific Application Group

Transaction command security
- Lets an application program issue a subset of IMS operator commands (using the DL/I CMD call)

Sign-on verification security
- Identifies a particular user to IMS to determine if each transaction entered by that user is authorized to the user id currently logged on

MVS users can also use the RACF program product if desired. See "Sign-on Verification Security" on page 183.

Although IMS system definition creates most resident control blocks for the IMS control program, it supplies only basic terminal security which prohibits the entry of certain commands from any terminal other than the master terminal. If no security options are specified by system definition, the generated IMS system protects the following commands from non-master terminal use:

/ASSIGN /DEQUEUE /MSASSIGN /RMxxxxxx
/CHANGE /DISPLAY /MSVERIFY /RSTART
/CHECKPOINT /ERESTART /NRESTART /SMCOPY
/CLSDST /IDLE /OPNDST /SSR
/COMPT /LOOPTEST /PSTOP /START
/DBDUMP /MODIFY /PURGE /STOP
/DBRECOVERY /MONITOR /QUIESCE /TRACE
/DELETE

Note:
1. RMLIST is not protected from non-master terminal use.

The basic level of security is called default terminal security. It exists whether or not the Security Maintenance utility is used to implement the added security features of IMS.
Input and Output Flow for DFSISMPO

Figure 51 on page 181 shows the flow of input to and output from the Security Maintenance utility. When you run the Security Maintenance utility, it receives input from the IMS.SDFSRESL data set, the IMS.MODBLKS data set, and from input statements. The utility outputs a security listing and a set of secured resource tables to the IMS.MATRIX data set. The output includes the following secured resource tables:

- Communication password table (CPT)
- Communication password matrix (CPM)
- Password offset list
- Communication matrix (CTM)
- Terminal offset list (CTL)
- Transaction matrix
- Transaction offset list
- Sign-on offset list
- Application group names table

Refer to Figure 51 on page 181 as you read the remaining sections of this chapter.
Restrictions for DFSISMPO

The following restrictions apply to the Security Maintenance utility.

- If you do not use RACF two of the options, sign-on verification and IMS Resource Access Security, require user exit routines that you must provide. IMS does not supply exit code except for sample user exits.

- The Security Maintenance utility does not provide LTERM, password, resource, transaction command, or sign-on verification security for terminals defined with the Extended Terminal Option (ETO) or for LU 6.2 devices. If your system uses ETO or LU 6.2 terminals, use RACF or an equivalent security product to provide terminal security functions.

- SMU cannot be used for transaction command security for DL/I ICMD or RCMD calls.

Related Reading: For more information about security for ETO and LU 6.2 terminals, see IMS Version 8: Administration Guide: System.

Security Options for DFSISMPO

This section provides a brief explanation of each of the five security options for the Security Maintenance utility.
Security Options

LTERM Security
This security option allows you to define a set of commands and transactions that are authorized for use by specified logical terminals. You can define a maximum of 32767 different patterns for these sets of commands and transactions, a pattern being a unique set of commands issued by one or more terminals. For example, two terminals issuing the same set of commands constitutes one pattern and two terminals issuing different sets of commands constitutes two patterns.

For multiple IMS systems, you can also specify a set of authorized transactions that can be passed to an IMS system using a logical data link. Terminal security requires no external user exit routines.

Password Security
This option specifies the use of a password with terminal input. Resources cannot be used unless a correct password is supplied. Password security can be used for:
• Transactions
• Commands

When an operator enters the /LOCK or /UNLOCK command to control the use of online resources, the use of several keywords can be protected by requiring a password. The resources that can be protected are:
• PTERM
• LTERM
• Application programs
• Databases

Transaction Command Security
You must use a security option to authorize an online application program to issue IMS commands using the DL/I CMD and GCMD calls. You make the authorization by associating the transaction that invokes the program with a list of commands that the program is designed to use. In the case of an automated operator program, you need to specify all the commands the program is designed to use.

Related Reading: For more information on the list of commands that programs can issue, see IMS Version 8: Command Reference.

Transaction command security requires no user exit routines.

IMS Resource Access Security
Resource access security prevents an IMS resource from being used by a dependent message region unless that resource has been authorized for use by the dependent region. It also prevents an unauthorized dependent message region from starting.

You specify, in the SECURITY macro during system definition, whether or not resource access security will be included and whether RACF or a user exit routine will be used for the authorization validation.

You define, through Security Maintenance, a list of IMS resources and assign a unique application group name (AGN) to the list. Security Maintenance places this AGN in a table the IMS system recognizes. IMS prevents the use of resources by a dependent message region unless that resource is listed in the AGN table.
You must either provide a user exit routine or use RACF to authorize the start of a dependent message region.

Sign-on Verification Security

You can prevent an unauthorized user from accessing the IMS system from a local or remote terminal by defining a list of terminals that require sign-on verification. Sign-on verification security identifies a user to IMS as being present from the /SIGN ON command entry until a /SIGN OFF command is entered.

You can define a maximum of 32767 terminals to require sign-on verification.

You can also restrict those authorized users (user IDs) to specific transactions. You do this by specifying, along with sign-on verification security, transaction authorization. As each transaction is entered from a terminal, either RACF or a user exit routine, or both, validate it for the user ID signed on. If the terminal from which the transaction is entered does not require sign-on verification, the transaction code is still checked by RACF or by the user exit routine.

Resource Access Control Facility (RACF)

RACF is an independent facility that runs under MVS operating systems. RACF and Security Maintenance provide the same functions as user exit routines and Security Maintenance: sign-on verification and resource access security.

Two RACF security variations, not supplied by user exit routines and Security Maintenance, are terminal-user security and resource class security (provided by the RACF APPL resource class). RACF terminal-user security prevents an online user from using a physical terminal unless the appropriate user ID is specified under a user profile for that terminal. (RACF terminal-user security and password security are somewhat different from the same functions provided by Security Maintenance and user exit routines.) Through the RACF keyword APPL, resource class security lets you define any resource in the IMS system, including the system itself, as a resource to protect.

Command Authorization Exit Routine

You can supply an exit routine to restrict the commands a user can issue from a terminal or from an AO application that uses the ICMD call.

Related Reading: For a sample user exit, DFSCCMD0, see [IMS Version 8: Customization Guide]

If the command authorization user exit is included in the IMS system, IMS calls the exit to check whether a user ID is eligible to issue a command. If the value coded for IMS execution parameter keyword AOIS is either A or C, DFSCCMD0 is called to check whether the AO application is eligible to use the ICMD call to issue the command.

IMS Application Resource Access Security

Application resource access security permits online programs to determine the access authority of the user requesting the transaction. This level of security is available by issuing an Authorization (AUTH) call in conjunction with RACF.
IMS Application Resource Access Security

You define (to RACF) the resources whose access will be restricted within the application program. These resources include terminals, spool readers, and data sets.

You can use the normal IMS security exit routines (such as the /SIGN ON exit routine and the Transaction Authorization exit routine) for additional control of the authorization process.

**Related Reading:** For more information on the Authorization (AUTH) call, see *IMS Version 8: Application Programming: Database Manager*.

SECURITY Procedure

The SECURITY procedure is created as a part of system generation and is placed into the IMS.PROCLIB procedure library by stage two of IMS system definition.

To run the security maintenance program, you must have previously defined an IMS control program using the value ALL, ON-LINE, NUCLEUS, CTLBLKS, or MODBLKS as the second sub-list entry of the SYSTEM operand of the IMSCTRL macro instruction. Two of the modules created during Stage 2 of IMS system definition are a directory of communication resources and a directory of database resources of the defined system. They are placed in the IMS.SDFSRESL and IMS.MODBLKS data sets, respectively.

**Requirement:** These directories and the security maintenance control statements are the required input for the Security Maintenance utility.

The security maintenance program runs as a three-step job. The first step (Step S) accepts the input control and data statements and checks them against the IMS system being maintained to ensure correct format and validity. If there are no errors in the first step, the second step (Step C), an operating system assembly, is performed. The third step (Step L) is a link-edit which takes the assembly output from Step C and creates the following:

- Password table
- Password matrix
- Password matrix relative pointer lists
- Terminal matrix
- Terminal matrix relative pointer lists
- Transaction command matrix
- Transaction command matrix relative pointer list
- Sign-on relative terminal list
- Application Group Name table

Depending on the input presented, there are a variable number of output load modules created. The maximum size of any generated matrix is:

\[ M = \frac{I \times R}{8} \]

where:

- \( I \) is the number of secured resources as shown in Table 13 on page 185 for each matrix:
Table 13. Matrix Secured Resources for Variable I

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Secured Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password</td>
<td>Passwords</td>
</tr>
<tr>
<td>Terminal</td>
<td>Logical terminals and linknames</td>
</tr>
<tr>
<td>Transaction Command</td>
<td>Transactions</td>
</tr>
</tbody>
</table>

**Restriction:** To produce a valid terminal matrix, the number of LTERMs, transactions, databases, and programs specified at system generation cannot exceed 65535, otherwise a DFS1913E will be received with a return code of 8.

**R** is the number of secured resources as shown in Table 14 for each matrix. If more than one secured resource is secured to the same set of securing resources, only one secured resource is counted in order to calculate the size of the matrix. For example, if one or more transactions and/or commands can be entered by the identical sub-lists of LTERMs, these transactions and/or commands are counted as a single secured resource.

Table 14. Matrix Secured Resources for Variable R

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Secured Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password</td>
<td>Commands, databases, LTERMs, programs, PTERMs, transactions</td>
</tr>
<tr>
<td>Terminal</td>
<td>Transactions, commands</td>
</tr>
<tr>
<td>Transaction Command</td>
<td>Commands</td>
</tr>
</tbody>
</table>

**M** is the total virtual storage requirement in bytes.

**PROC Statement**

Figure 52 shows the JCL for the SECURITY procedure. The parameters are described in “JCL Parameter Description” on page 186.

```plaintext
//    PROC  OPTN=UPDATE,IMS='0',SOUT=A,SYST2=,RGN=2048K
//S  EXEC  PGM=DFSISMP0,PARM='&OPTN;&IMS;'  
//STEPLIB  DD  DSN=IMS.&SYS2  MODBLKS,DISP=SHR
//        DD  DSN=IMS.&SYS2  SDFSRESL,DISP=SHR
```

Figure 52. JCL for the SECURITY Procedure (Part 1 of 2)
SECURITY Procedure

JCL Parameter Description

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 operand must be enclosed in quotes and must include a trailing period; for example, SYS2='IMS.A.'.

RGN=
Specifies the region size for this procedure. In Figure 52 on page 185 the region size is specified as 2048 KB.

Step S EXEC Statement

The EXEC statement specifies the program name (PGM=DFSISMP0), and must contain a PARM keyword value in the form:

```
//STEP  EXEC  PGM=DFSISMP0,PARM='option,number'
```

option
Is one of the following:

LIST
Validity check and list new security tables.

UPDATE
Validity check, list, and update security tables in MATRIX.

number
Is any valid alphanumeric that is the last character of the IMS nucleus member name to be maintained.
The default value for PARM is 'UPDATE,0'.

**DD Statements**

**STEPLIB**
Defines the partitioned data sets named IMS.SDFSRESL and IMS.MODBLKS. Contains the members DFSVNUCn, DFSISMP0, DFSISDBn, and DFSISDCn.

**SYSPRINT DD**
Defines a sequential message data set. The data set can be written to system output devices, magnetic tape, or direct-access volumes. The following DCB parameters must be specified:

- RECFM=VBA
- LRECL=125
- BLKSIZE=129 or greater

**SYSPUNCH DD**
Defines a sequential output data set that contains assembler statements produced by this step. The data set can be passed to Step C. The following DCB parameters must be specified:

- RECFM=F or FB
- LRECL=80
- BLKSIZE=80 or multiple of 80

**SYSLIN DD**
Defines a sequential output data set that contains linkage editor control statements produced by Step S. The data set can be passed to Step L. The following DCB parameters must be specified:

- RECFM=F or FB
- LRECL=80
- BLKSIZE=80 or multiple of 80

**SYSUT1 DD**
Defines a sequential work data set used only during this step. The following DCB parameters must be specified:

- RECFM=F or FB
- BLKSIZE=100 or multiple of 100

**SYSUT2 DD**
Defines a sequential work data set used only during this step. The following DCB parameters must be specified:

- RECFM=F or FB
- BLKSIZE=100 or multiple of 100

**SYSIN DD**
Defines a sequential data set or a member of a partitioned data set that contains security maintenance input statements. The following DCB parameters must be specified:

- RECFM=F or FB
- BLKSIZE=80 or multiple of 80

**Step C**

Step C is the assembly step.

**Related Reading**: Refer to *High Level Assembler Programmer’s Guide* for information on assembly steps.
SECURITY Procedure

**DD Statements**

**SYSIN DD**

Defines the passed sequential input data set created in steps using DDname SYSPUNCH.

**Step L**

Step L is the link-edit step.

**Related Reading:** Refer to *MVS/DFP Linkage Editor and Loader* for information on linkage-editors.

**DD Statements**

**SYSLMOD DD**

Defines an output partitioned data set, IMS.MATRIX, for the linkage editor.

**INPUT DD**

Defines the passed sequential temporary data set created using DDname SYSGO in Step C.

**Invoking the Procedure**

The following JCL is used to invoke the SECURITY procedure:

```plaintext
// JOB
/*
//STEP EXEC PROC=DFSISMP0
/*
//SYSIN DD *
:
END
*/
EXEC
   Should be in this form:
   //STEP EXEC PROC=DFSISMP0

**SYSIN DD**

Defines a sequential data set or a member of a partitioned data set that contains security maintenance input statements. The following DCB parameters must be specified:

- **RECFM=F or FB**
- **BLKSIZE=80 or multiple of 80**

**Utility Control Statements for DFSISMP0**

Input to the Security Maintenance utility consists of control and data statements. A control statement indicates to the utility that security is being established for the system resource named by that statement. Control statements are identified by closed and open parentheses characters in combination, )(), in character positions 1 and 2 of the input statement, followed by a blank in character position 3. A control statement remains in effect until another control statement or end of input data is encountered.

Data statements describe the security to be established for the system resource defined by the preceding control statement. All data statements following a control statement are associated with that control statement. Data statements are identified by a blank in character position 1 of the input statement. Each statement, control or
data, has only one operand, separated from the operation by a blank. Any number of valid data statements can be used in conjunction with a given control statement.

Input data must be entered in columns 1 through 71 of the input statements; columns 72 through 80 are ignored.

Comments can be included following the parameter specifications on control and data statements. Comment-only statements can be specified by an asterisk in column 1.

Use Table 15 to determine which input statements can be used as control statements, data statements, or both.

Table 15. Security Maintenance Utility Input Statements

<table>
<thead>
<tr>
<th>Input Statement</th>
<th>Control Statement</th>
<th>Data Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGN</td>
<td>X</td>
<td>N/A</td>
</tr>
<tr>
<td>SIGN</td>
<td>X</td>
<td>N/A</td>
</tr>
<tr>
<td>AGLTERM</td>
<td>N/A</td>
<td>X</td>
</tr>
<tr>
<td>AGPSB</td>
<td>N/A</td>
<td>X</td>
</tr>
<tr>
<td>AGTRAN</td>
<td>N/A</td>
<td>X</td>
</tr>
<tr>
<td>STERM</td>
<td>N/A</td>
<td>X</td>
</tr>
<tr>
<td>COMMAND</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>CTRANS</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>DATABASE</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>PASSWORD</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>PTERM</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>TCOMMAND</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>TERMINAL</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>TRANSACT</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

The operands which can be used with these statements are:

**Password**
A password is any combination of 1 to 8 alphanumeric characters. The longest password encountered on a PASSWORD statement in the input stream governs the maximum length of input passwords that will be accepted by your IMS system.

To define additional passwords, a PASSWORD control statement is used with no following data statements:

```plaintext
) ( PASSWORD ABCD
) ( PASSWORD EFGH
```

**Logical terminal name or linkname**
A valid logical terminal name or linkname is 1 to 8 characters in length. It must be defined in the IMS system being maintained or it is invalid. Any invalid terminal names or linknames are rejected by Security Maintenance.
Utility Control Statements

Transaction code
A valid transaction code is 1 to 8 characters in length. It must be defined in
the IMS system being maintained. If it is not defined, it is treated as invalid
by the utility.

Command language verb
Valid command language verbs are obtained from the Stage 2 output of
IMS system definition.

Related Reading: Refer to IMS Version 8: Installation Volume 2: System
Definition and Tailoring for more information on the command language
verbs used in Stage 2.

The command verb, less leading slash, can be abbreviated to the first three
characters.

Name Name is a valid database name, program name, VTAM® node name,
Application Group name, or BTAM physical terminal number as obtained
from the output of IMS system definition.

Only the first three characters of the operation code are required to identify control
or data statements.

Physical terminal numbers are found in the Stage 1 listing and in the assembly of
DFSISDB0 in Stage 2 of the IMS system definition.

The following list shows the valid combinations of control statements and data
statements that Security Maintenance accepts:

<table>
<thead>
<tr>
<th>Control Statements</th>
<th>Valid Data Statements</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGN</td>
<td>AGLTERM, AGPSB, AGTRAN</td>
</tr>
<tr>
<td>COMMAND</td>
<td>PASSWORD, TERMINAL</td>
</tr>
<tr>
<td>CTRANS</td>
<td>TCOMMAND</td>
</tr>
<tr>
<td>DATABASE</td>
<td>PASSWORD</td>
</tr>
<tr>
<td>PASSWORD</td>
<td>COMMAND, DATABASE, PROGRAM, PTERM, TERMINAL, TRANSACT</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>PASSWORD</td>
</tr>
<tr>
<td>PTERM</td>
<td>PASSWORD</td>
</tr>
<tr>
<td>SIGN</td>
<td>STERM, LINE, TERMINAL</td>
</tr>
<tr>
<td>TCOMMAND</td>
<td>CTRANS</td>
</tr>
<tr>
<td>TERMINAL</td>
<td>COMMAND, PASSWORD, TRANSACT</td>
</tr>
<tr>
<td>TRANSACT</td>
<td>PASSWORD, TERMINAL</td>
</tr>
</tbody>
</table>

Output for DFSISMPO

Output from the Security Maintenance utility consists of up to six sequential
members that are placed in IMS.MATRIX.

Restriction: These members cannot be reprocessed using the linkage editor.

The contents and names of the six members output from the Security Maintenance
Utility are shown in Table 16 on page 191.
<table>
<thead>
<tr>
<th>Contents</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication Password Table/Matrix</td>
<td>DFSISPBx</td>
</tr>
<tr>
<td>Terminal Offset List</td>
<td>DFSISTLx</td>
</tr>
<tr>
<td>Transaction Command Matrix</td>
<td>DFSISTCx</td>
</tr>
<tr>
<td>Transaction Offset List and Table</td>
<td>DFSISTTx</td>
</tr>
<tr>
<td>Sign-on Table</td>
<td>DFSISSOx</td>
</tr>
<tr>
<td>Application Group Name Table</td>
<td>DFSAGTOx</td>
</tr>
</tbody>
</table>

The utility also provides a listing of the maintenance tables created. Each run of the utility replaces previously created members. A set of security maintenance tables can be maintained for each IMS online control program nucleus. It is identified by the last character of the IMS nucleus name.

**Security-Status Reports**

Each execution of the utility produces a printed analysis of the IMS system for which security is being maintained. If errors are encountered in processing the input control statements, no security block update functions are performed. Instead, diagnostic error messages are produced for the entire input stream.

You can also request a no-update execution of the Security Maintenance utility to produce a printed analysis of your IMS system security specifications. This run, using the LIST option on the EXEC statement in Step S (under “Invoking the Procedure” on page 188) ends with a return code of 16.

**Examples of DFSISMPO**

The following examples show input statements for the Security Maintenance utility.

**Example 1**

This example illustrates passwords assigned to each program.

```plaintext
) ( PROGRAM ACCT
    PASSWORD DOLLAR

) ( PROGRAM ENG560
    PASSWORD PARTNO

) ( PROGRAM LOGREC
    PASSWORD NONE

) ( PROGRAM AGC0568
    PASSWORD MONEY
```

**Example 2**

This example illustrates passwords assigned to each database.

```plaintext
) ( DATABASE ACCTLOG
    PASSWORD LOG

) ( DATABASE ACCTREC
    PASSWORD REC

) ( DATABASE ACTIVITY
    PASSWORD ACTIVE

) ( DATABASE ENGREC
```
Examples

PASSWORD PIER SQ

)( DATABASE PARTSREC PASSWORD PIER SQ

)( DATABASE PARTSREC PASSWORD ASSY

Example 3

This example illustrates passwords assigned to commands.

)( COMMAND CHANGE PASSWORD PSWD1

)( COMMAND PURGE PASSWORD PSWD2

Example 4

This example illustrates passwords assigned to transaction codes and a list of terminals that can use each transaction code. With the list of terminals is the required password restricting some IMS terminal commands to the master terminal only.

)( TRANSACT ACCTCHG PASSWORD CHARGE TERMINAL A875111 TERMINAL C8751112 TERMINAL D8751113 TERMINAL A8751114 TERMINAL A8751115

)( TRANSACT ACTY PASSWORD GO TERMINAL A8751111 TERMINAL C8751112 TERMINAL D8751113 TERMINAL A8751114 TERMINAL A8751115

)( TRANSACT TNL PASSWORD QTY TERMINAL DEPT650 TERMINAL DEPT610 TERMINAL DEPT620 TERMINAL DEPT631 TERMINAL DEPT632 TERMINAL DEPT630 TERMINAL DEPT640 TERMINAL DEPT641 TERMINAL DEPT642

)( TRANSACT ING PASSWORD QUESTION TERMINAL DEPT310 TERMINAL DEPT311 TERMINAL DEPT312 TERMINAL DEPT410 TERMINAL DEPT411 TERMINAL DEPT412 TERMINAL DEPT510 TERMINAL DEPT511 TERMINAL DEPT512 TERMINAL DEPT100 TERMINAL DEPT200 TERMINAL DEPT686
Example 5

This example illustrates that the master terminal can enter a subset of IMS terminal commands and transaction codes defined by the system definition example in this manual.
Examples

COMMAND CHANGE
COMMAND DELETE
COMMAND ASSIGN
COMMAND CHECKPOINT
COMMAND DBDUMP
COMMAND NRESTART
COMMAND ERESTART
COMMAND DBRECOVERY
COMMAND IDLE
COMMAND RSTART
COMMAND DISPLAY

Example 6

This example illustrates a list of terminals that must enter a sign-on command to gain access to IMS.

\[
\begin{align*}
\text{SIGN} & \quad \text{STERM} \quad \text{ALL} \\
\text{SIGN} & \quad \text{STERM} \quad 4 \quad \text{STERM} \quad 09 \quad \text{STERM} \quad 105 \quad \text{STERM} \quad \text{VTAM} \quad \text{STERM} \quad \text{V3270} \\
\text{SIGN} & \quad \text{STERM} \quad \text{LINE} \quad 3 \quad \text{TERMINAL} \quad 4 \quad \text{STERM} \quad \text{LINE} \quad 5 \quad \text{TERMINAL} \quad 2
\end{align*}
\]

Example 7

This example illustrates the relating of transactions to commands.

\[
\begin{align*}
\text{CTRANS} & \quad \text{ADDPART} \\
\text{TCOMMAND} & \quad \ast \\
\text{TCOMMAND} & \quad \text{STOP} \quad \text{CTRANS} \quad \text{ADDINV} \quad \text{CTRANS} \quad \text{APOL11} \quad \text{CTRANS} \quad \text{APOL12} \\
\text{CTRANS} & \quad \text{APOL13} \quad \text{TCOMMAND} \quad \text{COMPT}
\end{align*}
\]

Example 8

This example illustrates a list of programs, transactions, and logical terminals whose access is restricted to the region associated with the AGN name.

\[
\begin{align*}
\text{AGN} & \quad \text{TEST005} \quad \text{AGPSB} \quad \text{DDLTBP01} \quad \text{AGTRAN} \quad \text{TRAN13C0} \quad \text{AGLTERM} \quad \text{DD3270L4} \\
\text{AGN} & \quad \text{TEST003} \quad \text{AGPSB} \quad \text{ALL} \quad \text{AGTRAN} \quad \text{ALL} \quad \text{AGLTERM} \quad \text{ALL} \\
\text{AGN} & \quad \text{TEST004} \quad \text{AGPSB} \quad \text{APOL1} \quad \text{AGPSB} \quad \text{A3270} \quad \text{AGPSB} \quad \text{GISBMP09} \quad \text{AGTRAN} \quad \text{ADDINV} \quad \text{AGTRAN} \quad \text{APOL15} \quad \text{AGLTERM} \quad \text{TERM0001}
\end{align*}
\]
Examples

AGLTERM TERM0001
AGLTERM TERM0002
AGLTERM TERM0003
AGLTERM TERM0004

) ( AGN TEST002
AGPSB A3270
AGTRAN ADDINV

) ( AGN TEST002
AGPSB INTCON

Chapter 5. Security Maintenance Utility (DFSISMP0)
Chapter 6. Online Change Utilities and Procedures

You can use Online Change to make changes to some IMS system resources without stopping the system. This chapter describes some of the utilities and procedures used to prepare the system for online changes.

In this Chapter:

- "Online Change Copy Utility (DFSUOCU0)"
- "INITMOD Procedure" on page 202
- "Global Online Change Utility (DFSUOLC0)" on page 204

Online Change Copy Utility (DFSUOCU0)

You must run the Online Change Copy utility 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. 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.

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. Issuing the Online Change command sequence to prepare and commit an online change causes the inactive library to become the active library. Using an MVS serialization service, this utility prevents other utilities from updating the staging (source) or inactive (target) libraries while the copy is in progress.

Related Reading: For more information on IMS.MODSTAT, see IMS Version 8 Installation Volume 1: Installation Verification

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
- **MATRIX** Control blocks for the system security tables
- **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 status data set, MODSTAT.

The same method of serialization prevents this utility from updating the active libraries while they are being used by an IMS online system.

### 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, MATRIXx, and MODBLKs data sets must be on shared non-duplex 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.
- 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 using this utility, only by using the Partition Definition utility. However, PSB related changes for HALDB can be made to the ACBLIBs, as long as no DBD changes are included.
- In an RSR environment, this utility has no effect on the copies of the ACBLIBx, FORMATx, MATRIXx, MODBLKSx, and MODSTAT data sets used on the tracking subsystem for tracking an active subsystem.

If the Online Change Copy utility is cancelled prior to completion, the status of the ACBLIB, FMTLIB, or MODBLKS data set is unpredictable. The data set 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 cancelled prior to its successful completion the data set is virtually useless.

### Online Change Copy Utility Procedure

Figure 53 on page 199 shows the OLCUTL statements used to invoke the Online Change Copy utility, OLCUTL PROC. This procedure is generated by system generation and is placed in the IMS.PROCLIB by stage two of system definition.

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.
The procedure statement must be in the following form:

```
PROC TYPE=,IN=,OUT=,SOUT=A,SYS=,SYS2=,OLCGLBL='DUMMY,',OLCLOCL=
```

These parameters are intended to be used to produce the OLCSTAT DD card or the MODSTAT/MODSTAT2 DD cards.

```
OLCGLBL='DUMMY,';OLCLOCL=
```

Produced by Stage 2 SYSGEN. Results in the OLCUTL proc being set up for local online change as the default with the following DD statements:

```
//MODSTAT DD DSN=IM.S.&SYS.MODSTAT,DISP=SHR
//MODSTAT2 DD DSN=IM.S.&SYS.MODSTAT2,DISP=SHR
//OLCSTAT DD DUMMY,DSN=IMSPLEX.OLCSTAT,DISP=OLD
```

If global online change is desired, set the OLCGLBL= and OLCLOCL= parms as follows:

```
OLCGLBL=,OLCLOCL='DUMMY,'
```

This will result in the following DD statements to be used for global online change:

```
//MODSTAT DD DUMMY,DSN=IM.S.&SYS.MODSTAT,DISP=SHR
//MODSTAT2 DD DUMMY,DSN=IM.S.&SYS.MODSTAT2,DISP=SHR
//OLCSTAT DD DUMMY,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 operand 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
  operand 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 is:
  PGM=DFSUOCU0,PARM=(copy_type,input_library,target_library)

  copy_type
  Specifies the library to be copied. It can be the ACB, FORMAT, MATRIX, or
  MODBLKS library.

  input_library
  Defines the library DDnames to be used as input.

  Parameter  Meaning
  S            IMS staging library (IMSACB, FORMAT MATRIX, or MODBLKS)
  I            User input library (IMSACBI, FORMATI, or MODBLKSI)

  The use of the I parameter will permit the use of an input library other than the
  staging library.

  target_library
  Defines the library DDnames to be used for output.

  Parameter  Meaning
  A            IMS A library (IMSACBA, FORMATA, MATRIXA, or
                MODBLKSA)
  B            IMS B library (IMSACBB, FORMATB, MATRIXB, or
                MODBLKSB)
  G            Target library (inactive) determined by the utility, using the
                OLCSTAT data set. The target will be the library not currently in
                use by the IMS online system.
  O            User output library (IMSACBO, FORMATO, MATRIXO, or
                MODBLKSO)
  U            Target library (inactive) determined by the utility, using the
                MODSTAT data set. The target will be the library not currently
                in use by the IMS online system.

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.

  The use of the O parameter will permit a target data set other than the active or
  inactive data set.

  Recommendation: Use of the U parameter is recommended for an IMS that
  supports local online change. G is recommended for an IMSplex that supports
  global online change.

DD Statements
IMSACB DD
IMSACBA DD
IMSACBB DD
  Defines the staging, active, or inactive ACBLIB.
FORMAT DD
FORMATA DD
FORMATB DD
   Defines the staging, active, or inactive MFS 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

MATRIX DD
MATRIXA DD
MATRIXB DD
   Defines the staging, active, or inactive library containing the security tables.

MODBLKS DD
MODBLKSA DD
MODBLKSB DD
   Defines the staging, active, or inactive system definition 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.

Invoking the Procedure
Figure 54 on page 202 shows the JCL statements used to copy the staging libraries
   to the active libraries.
INITMOD Procedure

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.

Stage two of system generation places the INITMOD procedure in the IMS.PROCLIB procedure library. Figure 55 shows the JCL required for the INITMOD procedure.

```
// PROC SYS=,SYS2=,SF=,SOUT=A
// INIT1 EXEC PGM=IEBGENER
// SYSPRINT DD SYSOUT=*&SOUT
// SYSUT2 DD DSN=IMS.&SYS.MODSTAT&SF,DISP=OLD;
// SYSIN DD DUMMY
// SYSUT1 DD DSN=IMS.&SYS2.PROCLIB(DFSMREC),DISP=SHR
```

Figure 55. JCL for the INITMOD Procedure

PROC Statement

The procedure statement must be in the form:

```
PROC SYS=,SYS2=,SF=,SOUT=A
```

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 operand 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 operand 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 at system generation and is placed in the IMS.PROCLIB procedure
library. 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
   Is the DDname for the active IMS.MODBLKSA (B) data set that contains the
IMS system definition output. This also means that the DDname for the active
IMS.MATRIXA (B) has the same suffix (MATRIXA).

   If MODBLKSA is specified for example, IMS assumes that MATRIXA is the
active IMS.MATRIXA (B) library used for security maintenance utility output.

IMSACBA
   Is the DDname in the IMS procedure for the active IMS.ACBLIBA(B) library.

FORMATA
   Is the DDname of the active IMS.FORMATA (B) data set which 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.

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.

Figure 56] shows initialization of the MODSTAT ID to 3, and the DDnames to
MODBLKSB, IMSACBA, and FORMATA.

//INITI EXEC  PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSUT2 DD DSN=IMS.MODSTAT,DISP=OLD
//SYSIN DD DUMMY
//SYSUT1 DD *
3,MODBLKSB,IMSACBA,FORMATA

Figure 56. IEBGENER Job
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.

**Global Online Change Utility (DFSUOLC0)**

You can use the Global Online Change utility to initialize, recreate, or unlock the OLCSTAT data set. 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 IMSs 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. (See “Online Change Copy Utility (DFSUOCU0)” on page 197.)

**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 IMSs 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 IMSs 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.

**Related Reading:** For more information about the OLCSTAT data set, see *IMS Version 8: Installation Volume 1: Installation Verification*.
JCL Requirements

The JCL shown in Figure 57 shows the statements used to invoke the DFSUOLC procedure. This procedure is generated by stage 2 of system definition and placed in the IMS.PROCLIB.

```
//PROC FUNC=,ACBS=,MDBS=,FMTS=,MDID=,PLEX=,SOUT=A
//STEP1 EXEC PGM=DFSUOLC0,PARM=(FUNC,&ACBS,&MDBS,&FMTS,&MDID,&PLEX)
//STEPLIB DD DSN=IMS.&SYS2.SDFSRESL,DISP=SHR
//SYSUDUMP DD SYSOUT=&SOUT
//OLCSTAT DD DSN=IMSPLEX.OLCSTAT,DISP=OLD
//SYSPRINT DD SYSOUT=&SOUT
//SYSIN DD DUMMY
```

Figure 57. DFSUOLC Procedure

The following JCL will run with the DFSUOLC procedure and invoke the utility.

```
//DFSUOLC0 JOB
//STEP1 EXEC DFSUOLC,FUNC=,ACBS=,MDBS=,FMTS=,MDID=,PLEX=,
//SYSIN DD *
/*
```

The format of the EXEC statement is:

```
PGM=DFSUOLC0,PARM=(FUNC=&FUNC,&ACBS,&MDBS,&FMTS,&MDID,&PLEX)
```

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 statement 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 IMSs 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 IMSs to add must be specified with the SYSIN DD card.

Add IMSs that are current with the online change libraries; for example, IMSs that are currently up.

**Attention:** If you add an IMS that is not current with the online change libraries, warm starting that IMS can fail.

**DEL** Delete one or more IMSs from the list of IMSs 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 IMSs 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 IMSs can be specified with the SYSIN DD statement. If no IMSs are specified with the SYSIN DD statement, the list of IMSs 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 IMSs 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 IMSs 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 IMSs in the IMSplex failed during an online change. Online change sets a lock field in the OLCSTAT data set to prevent other IMSs 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 IMSs 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 IMSs fail during an online change.

MDBS
Specifies the IMS JCL MODBLKS DD statement suffix for the active MODBLKS data set and the IMS JCL MATRIX DD statement suffix for the active MATRIX data set. The suffix can be A or B. A means MODBLKSA and MATRIXA are the DD statements of the active libraries. B means MODBLKSB and MATRIXB are the DD statements of the active libraries.

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 XCF CSL
IMSp lex group name for the UNL function. PLEX is required for the
UNL function. All OM, RM, SCI, IMS, etc., IMSp lex members that
are in the same IMSp lex sharing group sharing either data bases or
message queues must specify the same identifier. The same
identifier must also be used for the IMSPLEX= parameter in the
CSLSxxxx, CSLOlxxx, CSLRIxxx and DFSCGxxx proclip 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 IMSp lex.

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 IMSs to define, add,
or delete. Specify one IMSid per line.

The SYSIN DD statement specified with the ADD function adds one
or more IMSs to the existing list of IMSs in the OLCSTAT data set.

The SYSIN DD statement specified with the DEL function deletes
one or more IMSs from the existing list of IMSs in the OLCSTAT
data set.

The SYSIN DD statement specified with the INI function defines a
new list of IMSs. If IMS records already existed, they are wiped out.

Global Online Change Utility Examples

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,FMYS=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 IMSs 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 IMSs are current with the online change libraries, you could include those IMSs in the list. If IMSIDs are not specified, no IMSID will be listed on the OLCSTAT data set record.

```jcl
//DFSUOLC0 JOB
//STEP1 EXEC DFSUOLC,FUNC=INI,ACBS=A,MDBS=A,FMTS=A,MDID=0
//SYSIN DD
IMSA
IMSB
/*
//```
Chapter 7. Dynamic SVC Utility (DFSUSVCO)

The Dynamic SVC utility allows you to implement a new Type 2 SVC without a re-IPLing the MVS operating system. This is done by changing the MVS SVC Locator Table pointer from the old SVC (Nucleus resident) to a new SVC (Global Storage resident).

This utility runs as an MVS job.

In this Chapter:

- Restrictions for DFSUSVCO
- Input and Output for DFSUSVCO
- Error Processing for DFSUSVCO on page 210
- JCL Requirements for DFSUSVCO on page 210
- Example of DFSUSVCO on page 211

Restrictions for DFSUSVCO

The following restrictions apply to the Dynamic SVC utility:

- If you reply NO to both the REUSE and DELETE WTORs, you are telling the utility to do nothing. This will cause the old SVC directory table to remain allocated indefinitely (at lease until the next MVS IPL).
- The JOB JCL must contain a DFSRESLB DD statement that references an IMS SDFSRESL.
- The new IMS Type 2 SVC must be in an IMS SDFSRESL (standard IMS SYSGEN).
- The IMS SDFSRESL must reflect the correct SVC number that is to be replaced. This value is created by IMS SYSGEN and is located in the IMS Secondary System Contents Directory (SSCD). One possible way of introducing a user error is to point to the wrong library where a different SVC number (or even non-IMS SVC number) can be associated with this library. Check with your system administrator before using this utility.
- The IMS SDFSRESL that contains the SSCD and the new SVC 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).
- No IMS image (control region, batch, or utility) that uses this particular Type 2 SVC number can be active while attempting the dynamic update.

Input and Output for DFSUSVCO

The input to this utility is the new type 2 dynamic SVC module pointed to by the DSFSDFSRESL DD statement.

The utility takes this input and dynamically changes the entry of the MVS SVC Table (from the type 2 SVC in nucleus) to point to the new type 2 SVC which is resident in global storage.
Error Processing

Error Processing for DFSUSVCO

The following return codes are produced:

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SVC Table update was successful.</td>
</tr>
<tr>
<td>4</td>
<td>Cannot open DFSRESLB library.</td>
</tr>
<tr>
<td>8</td>
<td>DFSRESLB library not APF authorized.</td>
</tr>
<tr>
<td>12</td>
<td>DFSRESLB library does not contain module DFSVC000.</td>
</tr>
<tr>
<td>16</td>
<td>DFSRESLB library does not contain module IGCxxx, where xxx is the IMS SVC number.</td>
</tr>
<tr>
<td>20</td>
<td>Reply to WTOR was NO for message MVS SVC TABLE ENTRY WAS EMPTY.</td>
</tr>
<tr>
<td>24</td>
<td>Reply to WTOR was NO for message MVS SVC TABLE ENTRY CONTAINS AN ACTIVE SVC THAT IS NOT AN IMS SVC.</td>
</tr>
<tr>
<td>28</td>
<td>Reply to WTOR was NO for message MVS SVC TABLE ENTRY = IMS SVC AT HIGHER RELEASE THAT SDFSRESL SVC.</td>
</tr>
<tr>
<td>32</td>
<td>There is currently an active IMS using the SVC number in this SDFSRESL.</td>
</tr>
<tr>
<td>36</td>
<td>Nonzero return code from MVS SVC table update services (see WTO for MVS return code).</td>
</tr>
<tr>
<td>101</td>
<td>The module invoking the SVC must be running in supervisor mode.</td>
</tr>
</tbody>
</table>

JCL Requirements for DFSUSVCO

EXEC

Executes the Dynamic SVC utility, DFSUSVC0.

//STEP001 EXEC PGM=DFSUSVC0

DD Statements

STEPLIB DD
Points to an authorized library which contains the actual DFSUSVC0 utility (this should be in your IMS SDFSRESL).

//STEPLIB DD DSN=SOME.APF.AUTHORIZEDDATASET DISP=SHR

DFSRESLIB DD
Points to an authorized library which contains the new type 2 SVC module (input).

//DFSRESLIB DD DSN=SOME.IMS.SDFSRESL DISP=SHR

SYSPRINT DD
Specifies the output data set that is to contain the control messages. The DCB parameters for this data set are RECFM=FBA and LRECL=121. BLKSIZE can be provided on the SYSPRINT DD statement. If the BLKSIZE is not provided, a default value of 605 is used.

//SYSPRINT DD SYSOUT=*,
// DCB=(RECFM=FBA,LRECL=121,BLKSIZE=605),
// SPACE=(605,(10,10),RLSE,,ROUND)
Example of DFSUSVCO

```plaintext
//SVCINIT JOB MSGLEVEL=1,TIME=1440
//STEP001 EXEC PGM=DFSUSVCO
//STEPLIB DD DSN=SOME.APF.AUTHORIZED.DATASET,
// DISP=SHR
//DFSRESLB DD DSN=SOME.IMS.SDFSRESL,
// DISP=SHR
//SYSPRINT DD SYSOUT=*,
// DCB=(RECFM=FBA,LRECL=121,BLKSIZE=605),
// SPACE=(605,(10,10),RLSE,,ROUND)
```
Example
Part 3. Log Utilities

Chapter 8. Log Archive Utility (DFSUARC0) ................................................. 215
  OLDS Archive ................................................................................. 215
  Batch DASD Log Data Set Archive .................................................. 216
Optional Functions for DFSUARC0 ......................................................... 216
  Creating an RLDS (Recovery Log Data Set) .................................... 216
  Omitting Log Records on SLDS ......................................................... 217
  Copying Log Records into User Data Sets ...................................... 217
  Specifying User Exit Routines ......................................................... 217
  Specifying Forced End of Volume (EOV) ....................................... 217
Utility Input for DFSUARC0 ................................................................. 217
  OLDS Input .................................................................................... 217
  SLDS Input .................................................................................... 218
Utility Output for DFSUARC0 ............................................................... 218
JCL Requirements for DFSUARC0 ......................................................... 220
  DD Statements ............................................................................. 220
Utility Control Statements for DFSUARC0 ......................................... 222
  SLDS Statement ........................................................................... 222
  COPY Statement ......................................................................... 223
  EXIT Statement ........................................................................... 225
Error Processing for DFSUARC0 ......................................................... 225
Examples of DFSUARC0 .................................................................... 226
  Example 1 ..................................................................................... 226
  Example 2 ..................................................................................... 226

Chapter 9. Log Merge Utility (DFSLTMG0) .................................................. 229
Restrictions for DFSLTMG0 ................................................................. 229
Input and Output for DFSLTMG0 ......................................................... 229
  Controlling the Log Merge ............................................................ 229
  Control Statement Format ............................................................. 229
JCL Requirements for DFSLTMG0 ......................................................... 231
  DD Statements ............................................................................. 231

Chapter 10. Log Recovery Utility (DFSULTR0) ............................................. 233
OLDS Recovery ................................................................................ 234
SLDS Recovery ................................................................................ 234
Input for DFSULTR0 ........................................................................ 234
  Single Log Input .......................................................................... 234
  Dual Log Input ............................................................................. 235
Output for DFSULTR0 ....................................................................... 236
  Interim Log Error ID Record ........................................................ 237
  Error Block Listing (SYSPRINT) ..................................................... 237
  REP Mode Verification Messages .................................................. 239
  Dump of Data Record ................................................................... 239
  Active Region Messages ............................................................... 240
JCL Requirements for DFSULTR0 ......................................................... 241
  DD Statements ............................................................................. 241
Utility Control Statements for DFSULTR0 ......................................... 243
  CLS Mode—Close an OLDS from the WADS or NEXT OLDS .......... 243
  DUP Mode—Recover an OLDS or SLDS (Create an Interim Log) .... 244
  REP Mode—Recover an OLDS or SLDS (Create a New Log) ........... 245
  PSB Mode—Print Active PSBs Report ............................................ 246
Error Processing ............................................................................... 246
Examples of DFSULTR0 .................................................................... 247
Chapter 8. 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. The utility runs as an MVS batch job, and multiple log archive utility jobs can execute concurrently. When dual output is requested, the SLDS consists of primary and secondary data sets.

In this Chapter:

- "OLDS Archive"
- "Batch DASD Log Data Set Archive" on page 216
- "Optional Functions for DFSUARCO" on page 216
- "Utility Input for DFSUARC0" on page 217
- "Utility Output for DFSUARC0" on page 218
- "JCL Requirements for DFSUARC0" on page 220
- "Utility Control Statements for DFSUARC0" on page 222
- "Error Processing for DFSUARC0" on page 225
- "Examples of DFSUARC0" on page 226

OLDS Archive

The online IMS system writes log records to an OLDS in a wraparound fashion. When an OLDS is filled, it can be copied to an SLDS using the Log Archive utility. The SLDS can be on DASD or tape.

IMS notifies DBRC whenever an OLDS is either filled or closed or both. DBRC updates the RECON data set to indicate that the OLDS is available to be archived.

Using the Log Archive utility, you can archive multiple OLDSs to a single SLDS as long as the OLDSs being archived were created consecutively by IMS. The JCL supplied to the utility defines which and how many OLDSs are to be archived. The GENJCL facility of DBRC allows you to specify:

- Which OLDSs should be included in the created JCL
- That all OLDSs not yet archived should be included

If all the specified OLDSs are archived successfully, DBRC updates the RECON data set to indicate that the OLDSs are now available for reuse by the online system. If the Log Archive utility job fails, re-run it.

If you do not specify automatic archiving, you must create the JCL to run the utility. If you specified automatic archiving, IMS calls the DBRC GENJCL function to generate JCL for the utility when the specified number of OLDSs has been filled or closed.

If the DBRC JCLOUT DD statement for the GENJCL output is directed to the internal reader, the archive jobs are automatically started. Figure 58 on page 216 shows an overview of the Log Archive utility.
Batch DASD Log Data Set Archive

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 input data set can be either single or dual. 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.

Optional Functions for DFSUARCO

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 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.

Utility Input for DFSUARC0

The Log Archive utility has two types of input: OLDS and SLDS. The utility only accepts log data sets created by the same release of IMS/ESA 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
Utility Input

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 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.

Utility Output for DFSUARCC0

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

Figure 59 on page 219 is an example of a SYSPRINT listing of control statements.
Figure 60 is an example of a listing of checkpoint time stamp IDs. 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.

Figure 60. SYSPRINT Listing of Checkpoint Log Records

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.

Figure 61 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 ***

Figure 61. SYSPRINT Listing of Descriptive Messages

Figure 62 on page 220 shows an example of a SYSPRINT listing of the result of the archive.
JCL Requirements

*** 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 ............

Figure 62. Listing of the Result of the Archive

JCL Requirements for DFSUARC0

The Log Archive utility executes as a standard MVS job.

**Requirement:** A job statement, an EXEC statement, and DD statements that define input and output are required.

**EXEC**

Defines the utility to be run and optional execution parameters. Its format is:

```
//STEP EXEC PGM=DFSUARC0
   PARM= 'nnnn, DBRC=y, IMSPLEX=imsplex_name'
```

**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. See [IMS Version 8: Database Recovery Control (DBRC) Guide and Reference](https://www.ibm.com/support/knowledgecenter/ST45GL_8.4.0/com.ibm.zos.v8r4.cicdsc.doc/zos/orl/dbrcrclst.htm) for detailed information about the IMSPLEX parameter.

**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)

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 \textit{nn} (the suffix) is 00 through 99 and must be the same \textit{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 but they can be specified in any sequence in the DD statements. You can specify between 2 and 99 read buffers for the DCB BUFNO keyword.

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 \textit{nn} (the suffix) is 00 through 99 and must be the same \textit{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 but they can be specified in any sequence in the DD statements. 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.

Do not use the JCL parameter FREE=CLOSE on these DD statements. The data sets are dynamically deallocated and using FREE=CLOSE will 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.
JCL Requirements

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 for DFSUARC0

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:

```
/SLDS
NOLOG(\n)
FEOV(nnnnn)
/SLDS
```

NOLOG
Defines the log record types that are not to be copied to the SLDS. The value of a NOLOG sub-parameter 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.

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:

```
COPY DDNOUT1(nnnnnnnn) DDNOUT2(nnnnnnnn) RECORD
```

```
OFFSET(aaa) FLDTYPE VALUE(bbb) FLDLEN(ddd) COND
```

```
DBRECOV
```

The following abbreviations can be used in place of the keywords in the COPY
statement:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFFSET</td>
<td>O</td>
</tr>
<tr>
<td>FLDTYPE</td>
<td>T</td>
</tr>
<tr>
<td>VALUE</td>
<td>V</td>
</tr>
<tr>
<td>FLDLEN</td>
<td>L</td>
</tr>
<tr>
<td>COND</td>
<td>C</td>
</tr>
</tbody>
</table>

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.

**FLDYP(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 with X or in EBCDIC with C. 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).

**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 FLDYP(X)) of the VALUE field will be used. If FLDYP(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:

```
EXIT -- NAME(nnnnnnnn)
```

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 link-edited into either
JOBLIB or STEPLIB.

You can specify multiple EXIT control statements or multiple NAME parameters.

Error Processing for DFSUARC0

The Log Archive utility provides the following return codes:

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Archive processing completed successfully.</td>
</tr>
<tr>
<td>4</td>
<td>This return code is issued if one or both of the following events occur:</td>
</tr>
</tbody>
</table>
| | • 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. |
Error Processing

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.

Related Reading: Refer to [IMS Version 8: Messages and Codes, Volume 1] for descriptions of all error messages issued by DFSUARC0.

Examples of DFSUARC0

Example 1

The following example shows the JCL for the Log Archive utility using the COPY control statement to create an RLDS:

```jcl
//ARCHIVE JOB MSGCLASS=A,CLASS=A,MSGLEVEL=(1,1)
//ARC1 EXEC PGM=DFSUARC0,PARM='SYSA'
//STEPLIB DD DSNAME=IMS.SDFSRESL,DISP=SHR
// COPY FROM 3 OLDS TO A SLDS *
// RLDS AND A USER DATA SET ARE ALSO CREATED *
//DFSOLP00 DD DSNAME=OLP900,DISP=SHR,DCB=(BUFNO=20)
//DFSOLP01 DD DSNAME=OLP901,DISP=SHR,DCB=(BUFNO=20)
//DFSOLP02 DD DSNAME=OLP902,DISP=SHR
//DFSLOGP DD DSNAME=SLDSP.D82001.N001,DISP=(,KEEP),
//    UNIT=TAPE, VOL=(,,,99), LABEL=(,SL)
//RLDSD01 DD DSNAME=RLDSP.D82001.N001,DISP=(,KEEP),
//    UNIT=TAPE, VOL=(,,,99), LABEL=(,SL)
//USERD01 DD DSNAME=USER.D82001.N001,DISP=(,KEEP),
//    UNIT=3350, VOL=USER01, SPACE=(CYL,5)
//RECON1 DD DSNAME=RECON1,DISP=SHR
//RECON2 DD DSNAME=RECON2,DISP=SHR
//RECON3 DD DSNAME=RECON3,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//SYSIN DD *
// COPY DDNOUT1 (RLDSDD1) DBRECOV
// THIS USER DATA SET CONTAINS */
// COPY DDNOUT1 (USERD01) -
// RECORD (0(5) T(10) V(15) L(1) C(R)) -
// RECORD (0(5) T(10) V(16) L(1) C(R)) -
// RECORD (0(5) T(10) V(17) L(1) C(R))
EXIT NAME (UEXIT01)
```

Example 2

The following example shows the JCL for the Log Archive utility using FEOV to ensure consistency in the SLDS.

```jcl
//ARCHIVE2 JOB MSGCLASS=A,CLASS=A,MSGLEVEL=(1,1)
//*
//ARC2 EXEC PGM=DFSUARC0,PARM='SYSA'
//STEPLIB DD DSNAME=IMS.SDFSRESL,DISP=SHR
// COPY FROM 2 OLDS TO DUAL SLDS */
//DFSOLP02 DD DSNAME=OLP902,DISP=SHR
//DFSOLP00 DD DSNAME=OLP900,DISP=SHR
//DFSOL500 DD DSNAME=OLS900,DISP=SHR
//DFSOL502 DD DSNAME=OLS902,DISP=SHR
//DFSLOGP DD DSNAME=SLDSP.D82001.N001,DISP=(,KEEP),
//    UNIT=TAPE, VOL=(,,,99), LABEL=(,SL)
//DFSLOGS DD DSNAME=SLDSP.D82001.N001,DISP=(,KEEP),
//    UNIT=TAPE, VOL=(,,,99), LABEL=(,SL)
//RECON1 DD DSNAME=RECON1,DISP=SHR
```
Examples

//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. */
Examples
Chapter 9. Log Merge Utility (DFSLTMG0)

The Log Merge utility (DFSLTMG0) produces one data set that is used as input to the Log Transaction Analysis utility by merging the system log data sets (SLDS) from two or more IMS systems.

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.

In this Chapter:
- “Restrictions for DFSLTMG0”
- “Input and Output for DFSLTMG0”
- “JCL Requirements for DFSLTMG0” on page 231

Restrictions for DFSLTMG0

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.

Input and Output for DFSLTMG0

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. The utility only accepts input log data sets created by the same release of IMS as the utility release level.

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).

Restrictions: Do not use merged output as input to the Database Recovery utility.

Controlling the Log Merge

To control the log output:
- 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. 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.

Control Statement Format

START
Used to specify a start time. This statement must be present.
Input and Output

<table>
<thead>
<tr>
<th>Position</th>
<th>Length</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>START</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>blank</td>
</tr>
<tr>
<td>7</td>
<td>Variable</td>
<td>yyyddd,hhmmssstt(+,)HHMM where any trailing digits of hhmmssstt can be omitted and the optional time-zone information following hhmmssstt contains:</td>
</tr>
</tbody>
</table>

+ 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>
<thead>
<tr>
<th>Position</th>
<th>Length</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>STOP</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>blank</td>
</tr>
<tr>
<td>6</td>
<td>Variable</td>
<td>yyyddd,hhmmssstt(+,)HHMM where any trailing digits of hhmmssstt can be omitted and the optional time-zone information following hhmmssstt contains:</td>
</tr>
</tbody>
</table>

+ 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.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>All log record types are selected (this is the default if no control statements are present).</td>
</tr>
</tbody>
</table>
MSG
Selects all log records necessary for the Fast Path Log Analysis utility (DFSILTA0); X'01', X'03', X'06', X'07', X'08', X'3x' series, X'40', X'42', X'47'.

3X
Selects all log records within the range; X'30' to X'3F'.

XX
Where XX is the log record type selected.

JCL Requirements for DFSILMG0

**EXEC**
Executes the Log Merge utility DFSILMG0.

```
//STEP0 EXEC PGM=DFSILMG0
```

**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 DSNAME=IMS.LOGB,DISP=OLD,
// VOL=SER=XXXXXX,UNIT=TAPE
```

- **LOGOUT DD**
  Describes the output data set.

```
//LOGOUT DD DSNAME=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 *
```

**Example:** Sample control cards. This example will introduce an error code of 8. This occurs when the release level of the log input does not match the release level of the utility. Message DFS3062I indicates the reason.

```
START 75332,0830
STOP 75332,1030
MSG
```
JCL Requirements
Chapter 10. Log Recovery Utility (DFSULTR0)

You can 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. In a Remote Site Recovery (RSR) environment, do not use this utility on the tracking subsystem except in CLS mode to close the OLDS from the WADS.

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 exit routine (DFSFLGX0) is invoked during the execution of the Log Recovery utility if the exit routine is present. DFSFLGX0 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.

**Related Reading:** Refer to *IMS Version 8: Customization Guide* for a description of the Logger exit routine.

**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 DFS616l, which includes the LSN, at the point of failure. If DFS616l 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.

In an RSR environment, if you use this utility in any mode other than CLS, you can cause problems that might require you to reinstall the tracking subsystem.

If you lose a log volume on an active subsystem in an RSR environment, you might be able to get a copy from the tracking subsystem. However, consider this only as a last resort because the copy of the records might not be valid.

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

In this Chapter:

- [“OLDS Recovery” on page 234](#)
Log Recovery

- “SLDS Recovery”
- “Input for DFSULTR0”
- “Output for DFSULTR0” on page 236
- “JCL Requirements for DFSULTR0” on page 241
- “Utility Control Statements for DFSULTR0” on page 243
- “Error Processing” on page 246
- “Examples of DFSULTR0” on page 247

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 emergency restart, but must be closed using the Log Recovery utility when an emergency restart fails, or when the OLDS is not closed because a write error is detected.

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

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. The 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
- Log record sequence errors

Input for DFSULTR0

The Log Recovery utility uses both single and dual logs for input. The utility only accepts input log data sets created by the same release of IMS/ESA 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

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 can 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.

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. [Figure 63 on page 236](#) illustrates DUP mode and REP mode using dual logging.

Chapter 10. Log Recovery Utility (DFSULTR0) 235
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 for DFSULTR0

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

Figure 64 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.

![Diagram of error ID record on 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.

**CLS Mode and DUP Mode Error Listing**

```
ppppppppppppppppppp ON dddddddd BLOCK# bbbbbbb ** ERROR-ID=xnnnnn **
ssssssssssssssssssssssss--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.

**ddddd**
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.

**bbbbb**
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.

**nnnnn** 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 might be usable for restart or archive without doing so.

**ssssssssssssssssssssss**
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.

**fffffff TO ttttttt**
A block sequence, block TOD, or record sequence error has occurred. The variable ffffff is the last good value (or assumed good value). The variable
ttttttt 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 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 format of the printed output is shown in Figure 65 on page 240.
**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 format of this output is shown in [Figure 66](#).

---

### RECOVERY REQUIREMENTS

<table>
<thead>
<tr>
<th>PSB NAME</th>
<th>RECOVERY TOKEN</th>
<th>DATABASE</th>
<th>DSID</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPPPPPPP</td>
<td>EEEEEEEEEEHAAAAHHHHHHHHHHHHH</td>
<td>DDDDDDDD</td>
<td>NNN</td>
<td>MMMMMMMMM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SSSSSSSS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**END OF REPORT**

---

*Figure 66. Active Region Report*

The fields in the report have the following meanings:

- **PPPPPPP** The PSB name.
- **EEEEEEE** 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 DDDDDDD field, one of the following messages appears:
  - No database names found
  - DBNAME list 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.
The following JCL is required to run DFSULTR0. Examples of JCL using different modes appear in "Examples of DFSULTR0" on page 247.

EXEC
Invokes the Log Recovery utility (DFSULTR0). The format must be:

//STEP EXEC PGM=DFSULTR0,PARM='IMSID=iiiiiiii,
  //DBRC=ddd, IMSPLEX=imsplex_name'

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. See IMS Version 8: Database Recovery Control (DBRC) Guide and Reference for detailed information about the IMSPLEX parameter.

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.

Restriction: The following restrictions apply to the DD statements:

• 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.

• Specify OLDS input using the DFSOLP (and DFSOLS) DD statement.

• Specify SLDS input using the IEFRDER (and IEFRDER2) DD statement.
JCL Requirements

- 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.
- DFSPOLP (and DFSPOLS) DD statements cannot be specified 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.

See "Examples of DFSULTR0" on page 247 for valid DD statement combinations.

STEPLIB
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 chapter.

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.

DFSWADSn 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. This DD statement is used only when an OLDS is being closed from the WADS.

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 OLDSs are used.

IEFRDER DD
Defines the primary, or only, input SLDS. All input SLDS logs for DUP mode should have the same block size. (See "Example 9" on page 251 for
multivolume SLDS considerations when running DUP.) IEFDRDER 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.

**IEFDRDER2 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 for DFSULTR0**

**CLS Mode—Close an OLDS from the WADS or NEXT OLDS**

The format of this control statement is:

```
CLS NOWADS LSN=xxxxxxxxxxxxxxxxxxxx
```

**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 (see “Example 1” on page 247).
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 (see "Example 2" on page 247).

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, a
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—ERRC=nnnnn—LSN=xxxxxxxxxxxxxxxx

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 (see
"Example 9" on page 251). 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 (see "Example 3" on page 247 and "Example 5" on page 249).

**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 xxxxxxx, 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 xxxxxxxxxxxxxxx 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. (See "Example 4" on page 248 or "Example 6" on page 249.)

The format of the control statement is:

```
REP  SEQ=xnnnn  POS=ppppp  DAT=dd
```

- **REP**
  Indicates REP mode.
- **SEQ=xnnnn**
  Indicates the identification number of the block to be changed. The number is provided in the DUP mode listing output. See "Error Block Listing (SYSPRINT)" on page 237 for a description of the content of the error block listing.
- **POS=ppppp**
  Indicates the starting position, relative to 1, of the data being replaced.
- **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.

**DAT=dd**

```
  dd is 2 to 50 hexadecimal characters (0 through 9, A through F) representing the replacement data.
```

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.
Utility Control Statements

- 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 Active PSBs Report

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 (see “Example 7” on page 250 and “Example 8” on page 250). 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
```

**PSB**

Indicates PSB mode.

Error Processing

The Log Recovery utility provides the following return codes:

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion. If running CLS mode to terminate OLDS with WADS, ignore any error messages.</td>
</tr>
<tr>
<td>4</td>
<td>Successful completion—this condition code is accompanied by the following message: DFS3272I X'47' LOG RECORD NOT FOUND. ACTIVE PSB MESSAGES NOT GENERATED.</td>
</tr>
<tr>
<td>8</td>
<td>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.</td>
</tr>
</tbody>
</table>

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

**Related Reading:** Refer to the [IMS Version 8: Messages and Codes, Volume 1] for descriptions of all error messages issued by DFSULTR0.
Examples of DFSULTR0

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'
```
Examples

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).

See Example 4 for REP mode.

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.

See Utility Control Statements for DFSULTR0 on page 243 for an example of the formats of REP mode.
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.

See “Example 3” on page 247 for DUP mode.

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.

See “Example 6” for REP mode. See “Example 9” on page 251 for DUP mode with ERRC=00000.

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)
```
See "Utility Control Statements for DFSULTR0" on page 243 for an example of the formats of REP mode.

See "Example 5" on page 249 for DUP mode.

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.

//EXAMPLE07 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
/*
/*
/*
//SYSSIN 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.

//EXAMPLE08 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
    DD ......... OLDS or SLDS
/*
/*
//SYSSIN DD *
PSB

(see note below)
**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.

See "Example 5" on page 249 for DUP mode with ERRC=nnnnn.

See "Example 6" on page 249 for REP mode.

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 4. Analysis Utilities and Reports

### Chapter 11. IMS Monitor Report Print Utility (DFSUTR20)
- Restrictions for DFSUTR20 .................................. 257
- Input and Output for DFSUTR20 .......................... 257
- JCL Requirements for DFSUTR20 ......................... 257
  - DD Statements .............................................. 258
  - Analysis Control Data Set ............................... 258
  - Specifying Distribution Redefinition .................. 258
- Example of DFSUTR20 ..................................... 259

### Chapter 12. File Select and Formatting Print Utility (DFSERA10)
- Input and Output for DFSERA10 .......................... 261
- JCL Requirements for DFSERA10 ......................... 262
  - DD Statements .............................................. 262
- Utility Control Statements for DFSERA10 ............... 263
  - CONTROL Statement ...................................... 263
  - OPTION Statement ....................................... 264
  - Keywords .................................................. 266
  - END Statement ........................................... 269
  - COMMENTS Statement .................................... 269
- Examples of DFSERA10 .................................... 269
  - Example 1 .................................................. 269
  - Example 2 .................................................. 270
  - Example 3 .................................................. 270
  - Example 4 .................................................. 271
  - Example 5 .................................................. 272
  - Example 6 .................................................. 272
  - Example 7 .................................................. 273
  - Example 8 .................................................. 274
  - Example 9 .................................................. 274
- Record Format and Print Module (DFSERA30) ............ 275
  - The Deadlock Report .................................... 275
  - Utility Control Statements ............................. 281
- Program Isolation Trace Record Format and Print Module (DFSERA40) ... 282
  - DFSERA40 Utility Control Statements .................. 283
  - Output ..................................................... 283
- DL/I Call Image Capture Module (DFSERA50) ............. 286
  - Utility Control Statements ............................. 286
- IMS Trace Table Record Format and Print Module (DFSERA60) .... 286
  - Utility Control Statements ............................. 286
- Enhanced Select Exit Routine (DFSERA70) ................ 287
- Examples of Using the Enhanced Select Exit Routine (DFSERA70) .... 289
  - Example 1 .................................................. 289
  - Example 2 .................................................. 289
  - Example 3 .................................................. 289
  - Example 4 .................................................. 289
  - Example 5 .................................................. 289
  - Example 6 .................................................. 289
  - Example 7 .................................................. 289
  - Example 8 .................................................. 290

### Chapter 13. Fast Path Log Analysis Utility (DBFULTA0)
- Restrictions for DBFULTA0 ............................... 291
- Input and Output for DBFULTA0 .......................... 292
Chapter 11. 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).

In this Chapter:
- “Restrictions for DFSUTR20”
- “Input and Output for DFSUTR20”
- “JCL Requirements for DFSUTR20”
- “Example of DFSUTR20” on page 259

Restrictions for DFSUTR20

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 9999999 total DL/I calls if you plan to use Region Summary, Region Wait, Run-Profile, or Call-Summary (DB) reports. After 9999999 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).

Input and Output for DFSUTR20

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.

Related Reading: See Chapter 17, “Interpreting IMS Monitor Reports,” on page 347 for detailed information on output from the reports and an explanation of how to read them.

JCL Requirements for DFSUTR20

// JOB
   Initiates the job.

// EXEC
   Specifies the program name. The statement must be in the form:
JCL Requirements

// EXEC PGM=DFSUTR20,REGION=4096K

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. See “Specifying Distribution Redefinition” for information on how to redefine the distributions.

- 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 is 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 MVS 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 300000 3000000 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 Reading: Refer to [IMS Version 8: Administration Guide: System] for the default values for each distribution identifier and explanation of the distributions used by the various reports.

Example of DFSUTR20

The following JCL produces a complete set of reports, including the Call Summary report, from a tape with a serial number of IMSDA1.

```jcl
//TRACE JOB (969,6014),CHAPMAN,MSGLEVEL=(1,1),CLASS=A
//*
// EXEC PGM=DFSUTR20,REGION=512K
// STEPLIB DD DSN=IMS.SDFSRESL,DISP=SHR
// SYSPRINT DD SYSOUT=A
// SYSDUMP DD SYSOUT=A
// SYST1 DD DSN=IMSMON,DISP=(OLD,KEEP),
// UNIT=TAPE,VOL=SER=IMSDA1
// ANALYSIS DD *
DLI CALL REPORT
DISTRIBUTION */
```

If the distribution for D30 and D2 are modified, the JCL is modified as follows:

```jcl
//ANALYSIS DD *
DLI CALL REPORT
DISTRIBUTION
D30 8000,24000,50000,75000
D2 1000,2000,3000,4000,5000,6000,7000,8000,9000
/*
```
Example
Chapter 12. 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
- 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 exit routines 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.

In this Chapter:

- "Input and Output for DFSERA10"
- "JCL Requirements for DFSERA10" on page 262
- "Utility Control Statements for DFSERA10" on page 263
- "Examples of DFSERA10" on page 269
- "Record Format and Print Module (DFSERA30)" on page 275
- "Program Isolation Trace Record Format and Print Module (DFSERA40)" on page 282
- "DL/I Call Image Capture Module (DFSERA50)" on page 286
- "IMS Trace Table Record Format and Print Module (DFSERA60)" on page 286
- "Enhanced Select Exit Routine (DFSERA70)" on page 287
- "Examples of Using the Enhanced Select Exit Routine (DFSERA70)" on page 289

Input and Output for DFSERA10

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.
Input and Output

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 Requirements for DFSERA10

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**

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 SYSPUT=A.

DCB parameters specified for this data set are RECFM=FBA and LRECL=133. Block size can be provided on the SYSPRINT DD statement and must be a multiple of 133. The default is 133.

**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 for DFSERA10**

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 operand 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 operand 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 operand 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 COMMENTS statement to include any information in identifying tests or data. It has 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.
Utility Control Statements

CONTROL or CNTL
Identifies the CONTROL statement.

SKIP= or K=
Defines the first record tested. All prior records are ignored.

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.

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.

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 EXTR 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.
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.
Utility Control Statements

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. For a description of the possible parameters, see "Enhanced Select Exit Routine (DFSERA70)" on page 287.

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 FLDTYP=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 operand, 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.

&xxxxxxx

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.

**STARTAFT= 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.
Utility Control Statements

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 as described in 267 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.

COMMENTS Statement
The COMMENTS statement is optional. If used, the contents are displayed on the SYSPRINT data set.

An asterisk (*) entered at position 1 indicates a comment.

Examples of DFSERA10
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. Each example makes use of the COMMENTS statement to describe the functions being performed.

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 DSN=IMS.SDFSRESL,DISP=SHR
// SYSGEN DD SYSOUT=A
// SYSUT1 DD DSN=IMSLOG,DISP=(OLD,KEEP),
//         UNIT=TAPE,VOL=SER=123456, LABEL=(,SL)
// SYSUT4 DD DSN=EXAMPLE1.COPY1,DISP=(NEW,PASS),
//         UNIT=SYSDA,VOL=SER=IMSPAC,
//         SPACE=(TRK,(3,1))
// SYSGEN DD *
// **
*------------------------------------------------------------------*
* CONTROL STATEMENT : DEFAULTS *
* INPUT   = SYSUT1 *
* OUTPUT  = SYSGEN *
* SELECTION QUALIFIERS :
* 1. DEFAULT = ALL INPUT RECORDS *
*------------------------------------------------------------------*
OPTION PRINT END
*------------------------------------------------------------------*
* CONTROL STATEMENT : DEFAULTS *
* INPUT   = SYSUT1 *
* OUTPUT  = SYSUT4 *
Examples

* 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.

//EXAMPLE2 JOB
//*
// EXEC PGM=DFSELA10
// STEPLIB DD DSNNAME=IMS.SDFSRESL,DISP=SHR
// SYSPRINT DD SYSOUT=A
// LOGIN DD DSNNAME=IMSLOG,DISP=(OLD,KEEP),
//    UNIT=TAPE,VOL=SER=123456,LABEL=(,SL),
// SYSIN DD *
*-----------------------------------------------------------------*

CONTROL STATEMENT : SPECIFIED
* INPUT = LOGIN
* OUTPUT = SYSPRINT
* SELECTION QUALIFIERS :
*   1. TYPE X'16' IN 5TH BYTE = (ALL /SIGN ON/OFF) *
*-----------------------------------------------------------------*
CONTROL CNTL DNAME=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 DNAME=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=DFSELA10
// STEPLIB DD DSNNAME=IMS.SDFSRESL,DISP=SHR
// SYSPRINT DD SYSOUT=A
// LOGIN DD DSNNAME=IMSLOG,DISP=(OLD,KEEP),
//    UNIT=TAPE,VOL=SER=123456,LABEL=(,SL)
// LOGOUT DD DSNNAME=EXAMPLE3.COPY1,DISP=(NEW,PASS),
//    UNIT=SYSDA,VOL=SER=IMSPAC,
//    SPACE(TRK,(3,1))
// SYSIN DD *
*-----------------------------------------------------------------*

CONTROL STATEMENT : SPECIFIED
* INPUT = LOGIN
* OUTPUT = SYSPRINT
*-----------------------------------------------------------------*
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.

/*
Example 4 JOB
//*/
EXEC PGM=DFSER10
//STEPLIB DD DSNNAME=IMS.SDFRESL,DISP=SHR
//SYSPRINT DD SYSOUT=A
//LOGIN DD UNIT=TAPE,DISP=(OLD,KEEP),LABEL=(,SL),
//LOGIN DD DSNNAME=IMSLOG,DISP=(OLD,KEEP),
//LOGOUT DD DSNNAME=EXAMPLE4.COPY1,DISP=(NEW,PASS),
// SYSDA, VOL=SER=IMSPAC,
// SPACE=(TRK,(3,1))
//SYSIN DD *
-----------------------------------------------------------------*
* 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. *
*----------------------------------------------------------------*
Examples

OPTION COPY PRTSYS=Y,OFFSET=5,FLDTYP=X,VALUE=50,FLDLEN=1,COND=E
OPTION NEGOF OFFSET=12,FLDTYP=C,VALUE=USERBBBB,FLDLLEN=8,COND=M
OPTION COPY PRTSYS=Y,OFFSET=5,FLDTYP=X,VALUE=51,FLDLLEN=1,COND=E
OPTION NEGOF OFFSET=12,FLDTYP=C,VALUE=USERBBBB,FLDLLEN=8,COND=M
OPTION COPY PRTSYS=Y,OFFSET=5,FLDTYP=X,VALUE=52,FLDLLEN=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 DSNAME=IMS.SDFSRESL,DISP=SHR
// SYSPRINT DD SYSOUT=A
// SYST1 DD SYSNAME=IMSLOG,DISP=(OLD,KEEP),
// UNIT=TAPE,VOL=SER=123456,LABEL=(SL)
// LOGOUT1 DD DSNAME=EXAMPLE5.COPY1,DISP=(NEW,PASS),
// UNIT=SYSDA,VOL=SER=IMSPAC,
// SPACE=(TRK,(3,1))
// LOGOUT2 DD DSNAME=EXAMPLE5.COPY2,DISP=(NEW,PASS),
// UNIT=SYSDA,VOL=SER=IMSPAC,
// SPACE=(TRK,(3,1))
// LOGOUT3 DD DSNAME=EXAMPLE5.COPY3,DISP=(NEW,PASS),
// UNIT=SYSDA,VOL=SER=IMSPAC,
// SPACE=(TRK,(3,1))
// SYSIN DD *

*------------------------------------------------------------------*
* CONTROL STATEMENT : SPECIFIED *
* INPUT = DEFAULT (SYST1) *
* 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,FLDLLEN=8,COND=M
OPTION COPY PRTSYS=Y,OFFSET=5,FLDTYP=X,VALUE=16,FLDLLEN=1,COND=E
END

CONTROL CNTL DDNOUT=LOGOUT2
OPTION COPY OFFSET=9,FLDTYP=C,VALUE=USERBBBB,FLDLLEN=8,COND=M
OPTION COPY PRTSYS=Y,OFFSET=5,FLDTYP=X,VALUE=16,FLDLLEN=1,COND=E
END

CONTROL CNTL DDNOUT=LOGOUT3
OPTION COPY OFFSET=9,FLDTYP=C,VALUE=USERCCCC,FLDLLEN=8,COND=M
OPTION COPY PRTSYS=Y,OFFSET=5,FLDTYP=X,VALUE=16,FLDLLEN=1,COND=E
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 DSNAME=IMS.SDFSRESL,DISP=SHR
// SYSPRINT DD SYSOUT=A

272 Utilities Reference: System
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 DSNAM=IMS.SDFSRESL,DISP=SHR
// SYSPRINT DD SYSOUT=A
// LOGIN DD DSNAM=IMSLOG,DISP=(OLD,KEEP),
// UNIT=TAPE,Vol=SER=123456,LABEL=,(SL)
// SYSSIN DD *

****************************************************************
* CONTROL STATEMENT : SPECIFIED *
* INPUT = LOGIN *
* 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 (RECORD NO. 158) *
****************************************************************

CONTROL_CNTL STOPAFT=(1,E),DDNAME=LOGIN
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

****************************************************************
* CONTROL STATEMENT : DEFAULTS *
* INPUT = SYSSIN *
* 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

Examples Chapter 12. File Select and Formatting Print Utility (DFSERA10) 273
Examples

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
//SYSUT1 DD DSN=IMS.EXTERNAL.TRACE,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
*-----------------------------------------------------------------*
* CONTROL STATEMENT : SPECIFIED *
* INPUT = SYSUT1 *
* OUTPUT = SYSPRINT *
* SELECTION QUALIFIERS :
* 1. Log record type '6FA' in fifth and sixth byte = (all trace log records)
*-----------------------------------------------------------------*
CONTROL CNTL SKIP=0
OPTION PRINT OFFSET=5,FLDLEN=2,VALUE=67FA,COND=E,E=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 DSN=IMS.SDFSRESL,DISP=SHR
//SYSPRINT DD SYSOUT=A
//LOGIN DD DISP=SHR,DSN=IMSLOG
//SYSIN DD *
CONTROL CNTL DD=LOGIN
OPTION PRINT O=5,T=X,V=67FF,L=2,C=M
OPTION PRINT O=29,T=X,V=F0F8F3F2,L=4,C=E,E=DFSERA30
END
```
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 MVS dump format. It is an exit routine to 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 can be present on your system (see CVT field CVTLSO). Thus, the time printed by DFSERA30 might be different from the time reported by MVS when the record was written. The difference is equal to the leap second adjustment amount.

The Deadlock Report

The deadlock report contains information about the resources and resource owners for the following:

- Deadlocks resulting from 777 and 123 pseudoabends.
- 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.

Figure 67 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.
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 a 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 a FP database is slightly more complex. The lock name of a FP database is broken down as shown in Table 17:

<table>
<thead>
<tr>
<th>Byte Position</th>
<th>Lock Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lock ID</td>
</tr>
<tr>
<td>2-4</td>
<td>Relative Byte Address</td>
</tr>
<tr>
<td>5-6</td>
<td>DMCB Number</td>
</tr>
<tr>
<td>7</td>
<td>Area Number</td>
</tr>
<tr>
<td>8</td>
<td>Fast Path ID=C6</td>
</tr>
</tbody>
</table>

In a 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 three bytes displayed indicate the 30 bit RBA. You must multiply the displayed RBA by 4 to get the true RBA.
If the lock occurred at the CI level, the first two digits indicate the code X'00'. In this case, the next three 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 a FP database, the last two digits (Byte 8) of the lock name display the code "C6."

For example, the lock name 80000C02800101C6 occurred in a 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 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 pertains only to the waiter. This information is not available for the holder of the lock.

The current DL/I call being processed is reported as one of the following.
Record Format and Print

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>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.</td>
</tr>
<tr>
<td>REPL</td>
<td>DL/I call was REPL.</td>
</tr>
<tr>
<td>ISRT</td>
<td>DL/I call was ISRT or ASRT.</td>
</tr>
<tr>
<td>DLET</td>
<td>DL/I call was DLET.</td>
</tr>
<tr>
<td>POS</td>
<td>DL/I call was POS call on MSDB.</td>
</tr>
</tbody>
</table>

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.

In Figure 67 on page 276, 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.

Related Reading: See [IMS Version 8: Administration Guide: Database Manager](#) for a description of locking.

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:

- **GBID**: Get a block lock. Block level sharing only.
- **GZID**: 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**: 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**: Get a lock on the root of a database record.
- **GQCM**: 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**: Get a segment lock for a dependent segment. This is not used when IRLM is the lock manager.
- **GFPLL**: Get a Fast Path lock.

The Lock States

The lock state is the type or level of lock and is usually 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 eleven, 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

Table 18 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 18. PI Lock Compatibility Matrix

<table>
<thead>
<tr>
<th>Requested Level</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Held at 1</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>X</td>
</tr>
<tr>
<td>Held at 2</td>
<td>C</td>
<td>C</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Held at 3</td>
<td>C</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Held at 4</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

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 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. Table 19 is the resultant state matrix.

Table 19. IRLM Resultant State Matrix

<table>
<thead>
<tr>
<th>Requested Level</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Held at 1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>Held at 2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>Held at 3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>Held at 4</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>Held at 5</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>Held at 6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>Held at 7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>Held at 8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

The compatibility matrix is shown in Table 20 with compatibility indicated by a “C” and incompatibility by an “X”.
Table 20. IRLM Compatibility Matrix

<table>
<thead>
<tr>
<th>Requested Level</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Held at 1</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>X</td>
</tr>
<tr>
<td>Held at 2</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Held at 3</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Held at 4</td>
<td>C</td>
<td>C</td>
<td>X</td>
<td>C</td>
<td>C</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Held at 5</td>
<td>C</td>
<td>C</td>
<td>X</td>
<td>C</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Held at 6</td>
<td>C</td>
<td>C</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Held at 7</td>
<td>C</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Held at 8</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

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. In Figure 67 on page 276, the locks had the private attribute.

**Special Situations**
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 of 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.

**Related Reading:** See [IMS Version 8: Diagnosis Guide and Reference](#) for more information about these macros.

**A Reporting Anomaly**
There is one deadlock situation where the report is different.

LOCK 1
PST 1 owns STATE SHR
PST 3 waits STATE UPD
PST 2 waits STATE SHR

LOCK 2
PST 2 owns STATE UPD
PST 3 waits STATE UPD
PST 1 waits STATE UPD

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 HOLDER.

**Selecting Only the Deadlock Block**
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. Figure 68 contains the DFSERA10 control statements to select only the deadlock element from any pseudoabend snap.

```language
//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,FLDTYP=X,VALUE=67FF,COND=M
OPTION PRINT OFFSET=33,FLDLEN=8,FLDTYP=C,VALUE=DEADLOCK,COND=E,
X EXTR=DFSERA30
END /*
```

*Figure 68. Sample DFSERA10 Control Statements for Deadlock Element*

**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. Figure 69 is an example of the DFSERA30 report for a deadlock detected by an external subsystem.

```plaintext
PSEUDO ABEND RECORD ABEND NO = 0777 RECN = 000000162 TIME 12:24:07.1 DATE 1998.292
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: E2E8E2F340404040000000050000000
```

*Figure 69. Deadlock Report for External Subsystem-Detected Lock*

**Utility Control Statements**
Figure 70 on page 282 shows the control statements required to format type X'67' log records using the DFSERA30 exit routine.
Record Format and Print

<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 10</th>
<th>Column 16</th>
<th>72</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTROL</td>
<td>CNTL</td>
<td>OFFSET=5, FLDELEN=2,</td>
<td>X</td>
</tr>
<tr>
<td>OPTION</td>
<td>PRINT</td>
<td>VALUE=67aa, COND=E,</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXITR=DFSERA30</td>
<td></td>
</tr>
<tr>
<td>END</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Figure 70. Control Statements Required for DFSERA30*

where aa is the subrecord type and

- **aa=01** Specifies TRACE subrecord type
- **aa=FD** Specifies SNAP subrecord type
- **aa=FF** Specifies ABEND subrecord type

*Figure 71* 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 | 0093660 | 0AE9350 | 0AE92B0 | 0091E90 | 0AE991C | 17000000 | 7F000000 |
| AE9024 | 000020 | 80000000 | 520821CE | 008229C | 00820C6 | 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 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| AE90E4 | 0000E0 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| AE9104 | 000100 | 02F30C41 | 93179101 | 0502F004 | F30C4193 | 17F10441 | 93170000 | 00000000 | 00B66218 |

*Figure 71. Sample Formatted Log Print from DFSERA30*

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.

**Diagnosis, Modification, or Tuning 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.

**End of Diagnosis, Modification, or Tuning Information**

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.

**DFSERA40 Utility Control Statements**

Figure 72 shows the control statements required for DFSERA40.

```
<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 10</th>
<th>Column 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTROL</td>
<td>CNTL</td>
<td></td>
</tr>
<tr>
<td>OPTION</td>
<td>PRINT</td>
<td>OFFSET=5,FLDLEN=2,VALUE=67FA, X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COND=E,EXITR=DFSERA40</td>
</tr>
<tr>
<td>END</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

*Figure 72. Control Statements Required for DFSERA40*

**Output**

Figure 73 on page 284 is a sample output from DFSERA40. The spacing of fields is altered.
Figure 73. Sample Output from DFSERA40

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 HH:MM:SS.UUU, where UUU is milliseconds, relative to the date on which tracing started. If the return code
**Program Isolation Trace Record**

(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.
- **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.

**COMMENTS**
Specifies 'DL/I CALL' if a trace is requested from DFSDLA00. Other comments are for LMGR traces.
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 examples of DFSERA10 input control statements in the SYSIN data set can be used to retrieve DL/I call image capture data from the log data set.

Print all DL/I call image capture records:

```
Column 1  Column 10
OPTION  PRINT OFFSET=5,VALUE=5F,FLDTYP=X
```

Print selected DL/I call image capture records by PSB name call:

```
Column 1  Column 10
OPTION  PRINT OFFSET=5,VALUE=5F,COND=M
OPTION  PRINT OFFSET=25,FLDTYP=C,FLDLEN=8,VALUE=psbname,COND=E
```

Format DL/I call image capture records (in format acceptable as input for the DL/I test program DFSDDLTO):

```
Column 1  Column 10
OPTION  PRINT OFFSET=5,VALUE=5F,COND=M
OPTION  PRINT EXITR=DFSERA50,OFFSET=25,FLDTYP=C,FLDLEN=8,VALUE=psbname,COND=E
```

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) is an exit routine. It 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**

Figure 74 on page 287 shows the control statements required to invoke DFSERA60.
Use the Enhanced Select exit routine (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.
- 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). For information about the syntax of the OPTION statement, see "OPTION Statement" on page 264. The possible subparameters of PARM= are:

**XFMT=**

- Extends the X'50' log record format to enhance the retrievability of certain data entries.
- 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**

DSHRDSN XXXXXXXX DSHRLSN XXXXXXXXXXXX DSHRUSID
XXXXXXXXXX RACF-UID CCCCCCCC XXXXXXXXXXXXXXX

**XRF buffer and lock tracking**

TRAKPLSZ XXX 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**
Enhanced Select Exit Routine

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 of Using the Enhanced Select Exit Routine (DFSERA70)**

**Example 1**

Print all records including X'50'/X'5950' database records and expand the data in the X'5050' records.

```plaintext
OPTION PRINT EXITR=DFSERA70
```

**Example 2**

Print only X'50' database records with expanded data.

```plaintext
OPTION PRINT O=5,V=50,EXITR=DFSERA70
```

**Example 3**

Print X'50' database records with expanded data and 67 diagnostic records.

```plaintext
OPTION PRINT O=5,V=67,EXITR=DFSERA30
OPTION PRINT O=5,V=50,EXITR=DFSERA70
```

**Example 4**

Print all records in regular format including X'50'/X'5950' database records for a PST number of X'A' using a PSB named APPLPSB.

```plaintext
OPTION PRINT EXITR=DFSERA70,PARM=(XFMT=N,PST=X'A',PSB=APPLPSB)
```

**Example 5**

Print all records in regular format including X'50'/X'5950' database records at an RBA of X'2000' and an offset of X'200'.

```plaintext
OPTION PRINT EXITR=DFSERA70,PARM=(XFMT=N,RBA=X'2000',OFFSET=X'200')
```

**Example 6**

Print, in extended format, all records that contain the character string 'aaaa'.

```plaintext
OPTION PRINT EXITR=DFSERA70,PARM=(XFMT=Y,DATA=aaaa)
```

**Example 7**

Select all types of log records with the token X'0001F8FF00000000' and print the records in extended format.

```plaintext
OPTION PRINT EXITR=DFSERA70,PARM=(XFMT=Y,TOKEN=X'0001F8FF00000000')
```
Example 8

Select X'0A' log records with the token X'0001F8E400000001' and print the records in regular format.

OPTION PRINT O=5,V=0A,T=X,EXIT=DFSERA70,PARM=(XFMT=N,TOKEN=X'0001F8E400000001')
Chapter 13. Fast Path Log Analysis Utility (DBFULTA0)

Use the Fast Path (FP) Log Analysis utility to prepare statistical reports for Fast Path, based on data recorded on the IMS system log. This utility is an offline utility and produces three data sets, one of which contains six 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 trouble shooting. This utility is not related to the IMS Monitor or the Log Transaction Analysis utility.

Figure 75 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 75. 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, then 9999 will be displayed on 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 may be reported are:

- Input message (X’5901’) log record
Fast Path Log Analysis

- The routing code for the transaction
- The input terminal’s LTERM name for the transaction
- The balancing group queue count
- 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), and (D), and the performance-related items, are combined with other logged information to produce all the reports.

In this Chapter:

- “Restrictions for DBFUTLA0”
- “Input and Output for DBFUTLA0”
- “Detail-Listing-of-Exception-Transactions Report” on page 294
- “Summary-of-Exception-Detail-by-Transaction-Code (for IFP Regions) Report” on page 299
- “Overall Summary of Resource Usage and Contentions for All Transaction Codes and PSBs Report” on page 300
- “Summary-of-Region-Occupancy Report” on page 302
- “Summary-of-VSO-Activity Report” on page 303
- “Recapitulation-of-the-Analysis Report” on page 304
- “JCL Requirements for DBFUTLA0” on page 305
- “Utility Control Statements for DBFUTLA0” on page 306
- “Error Processing for DBFUTLA0” on page 311

Restrictions for DBFUTLA0

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.

Input and Output for DBFUTLA0

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.

**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 75 on page 291). 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.

[Figure 76 on page 295] is an example of a Detail-Listing-of-Exception-Transactions report.
### Exception Transactions Report

**Legend**

- **RT**: Region Type; I=IPF, M=MPP, B=BMP, D=DBCTL, U=UTILITY
- **PT**: Process Type; H=HSSP, R=REORG
- **CONTENTS**: C1: No. of WAITs for C1(S)
- **S**: No. of WAITs for COMMON Buffers
- **BF**: Sync Failure Codes - See Utility Reference Manual
- **BUF USE**: Total Buffers Used from the Common Pool - Includes NBA, OBA and NBDB (Non-Related Buffers for DBF/MBU Use)

**Detail Listing of Exception Transactions:**

| SEQ | TRANCODE | TIME | TRANCODE | TRANCODE | INDEX | COUNT | ID | ROW | TOTAL | BEGIN | RELAY | DBNAME | BSDNAME | EXEC | ROUTING | LOGICAL | PST | QUEUE | TRANSIT TIMES (MSEC) | IN-Q | PROC OUTQ | TOTAL (SEC) | END | CALL | RD | UPO | RD | UPO | CALL | USE | CI | WS | DW | T |
|-----|----------|------|----------|----------|-------|-------|----|-----|-------|-------|-------|--------|---------|------|----------|--------|----|-------|---------------|-----|---------|------------|----|-----|---|----|---|----|-----|---|----|---|
| 9   | TPCA     | 3:55:38.00 | TPCA     | FPT05505 | 52    | 105   | 68 | 20 | 45   | 133   | 0.1   | 5     | 3     | 1     | 4     | 2     | 0     | 5     | 0     | 0     | 0   | 1    | 1     | 1    | 0   | 1   | 0   | 0   | 0   | 1   |
|     |          |       |          |          |       |       |    |    |      |       |       |       |       |       |       |       |      |      |      |      |     |     |     |     |     |     |     |    |
|     |          |       |          |          |       |       |    |    |      |       |       |       |       |       |       |       |      |      |      |      |     |     |     |     |     |     |     |    |
| 58  | TPCA     | 3:55:38.03 | TPCA     | FPT04203 | 8     | 113   | 73 | 38 | 41   | 152   | 0.1   | 5     | 3     | 1     | 4     | 2     | 0     | 5     | 0     | 0     | 0   | 1    | 1     | 1    | 0   | 1   | 0   | 0   | 0   | 1   |
|     |          |       |          |          |       |       |    |    |      |       |       |       |       |       |       |       |      |      |      |      |     |     |     |     |     |     |     |    |
|     |          |       |          |          |       |       |    |    |      |       |       |       |       |       |       |       |      |      |      |      |     |     |     |     |     |     |     |    |
| 1   | TPCA     | 3:55:38.00 | TPCA     | FPT07383 | 64    | 107   | 66 | 38 | 48   | 152   | 0.1   | 5     | 3     | 1     | 4     | 2     | 0     | 5     | 0     | 0     | 0   | 1    | 1     | 1    | 0   | 1   | 0   | 0   | 0   | 1   |
|     |          |       |          |          |       |       |    |    |      |       |       |       |       |       |       |       |      |      |      |      |     |     |     |     |     |     |     |    |
|     |          |       |          |          |       |       |    |    |      |       |       |       |       |       |       |       |      |      |      |      |     |     |     |     |     |     |     |    |
| 25  | TPCA     | 3:55:38.01 | TPCA     | FPT07447 | 46    | 104   | 70 | 23 | 36   | 129   | 0.1   | 5     | 3     | 1     | 4     | 2     | 0     | 5     | 0     | 0     | 0   | 1    | 1     | 1    | 0   | 1   | 0   | 0   | 0   | 1   |
|     |          |       |          |          |       |       |    |    |      |       |       |       |       |       |       |       |      |      |      |      |     |     |     |     |     |     |     |    |
|     |          |       |          |          |       |       |    |    |      |       |       |       |       |       |       |       |      |      |      |      |     |     |     |     |     |     |     |    |
| 92  | TPCA     | 3:55:38.06 | TPCA     | FPT05963 | 47    | 127   | 72 | 29 | 43   | 144   | 0.1   | 5     | 3     | 1     | 4     | 2     | 0     | 5     | 0     | 0     | 0   | 1    | 1     | 1    | 0   | 1   | 0   | 0   | 0   | 1   |
|     |          |       |          |          |       |       |    |    |      |       |       |       |       |       |       |       |      |      |      |      |     |     |     |     |     |     |     |    |
|     |          |       |          |          |       |       |    |    |      |       |       |       |       |       |       |       |      |      |      |      |     |     |     |     |     |     |     |    |
| 88  | TPCA     | 3:55:38.06 | TPCA     | FPT00939 | 15    | 124   | 67 | 50 | 45   | 162   | 0.1   | 5     | 3     | 1     | 4     | 2     | 0     | 5     | 0     | 0     | 0   | 1    | 1     | 1    | 0   | 1   | 0   | 0   | 0   | 1   |
|     |          |       |          |          |       |       |    |    |      |       |       |       |       |       |       |       |      |      |      |      |     |     |     |     |     |     |     |    |
|     |          |       |          |          |       |       |    |    |      |       |       |       |       |       |       |       |      |      |      |      |     |     |     |     |     |     |     |    |
| 150 | TPCA     | 3:55:38.09 | TPCA     | FPT02509 | 24    | 111   | 77 | 30 | 35   | 142   | 0.1   | 5     | 3     | 1     | 4     | 2     | 0     | 5     | 0     | 0     | 0   | 1    | 1     | 1    | 0   | 1   | 0   | 0   | 0   | 1   |
|     |          |       |          |          |       |       |    |    |      |       |       |       |       |       |       |       |      |      |      |      |     |     |     |     |     |     |     |    |
|     |          |       |          |          |       |       |    |    |      |       |       |       |       |       |       |       |      |      |      |      |     |     |     |     |     |     |     |    |
| 148 | TPCA     | 3:55:38.09 | TPCA     | FPT02570 | 13    | 110   | 79 | 40 | 37   | 156   | 0.1   | 5     | 3     | 1     | 4     | 2     | 0     | 5     | 0     | 0     | 0   | 1    | 1     | 1    | 0   | 1   | 0   | 0   | 0   | 1   |
|     |          |       |          |          |       |       |    |    |      |       |       |       |       |       |       |       |      |      |      |      |     |     |     |     |     |     |     |    |
|     |          |       |          |          |       |       |    |    |      |       |       |       |       |       |       |       |      |      |      |      |     |     |     |     |     |     |     |    |

**Figure 76. Sample Detail Listing of Exception Transactions**

The column headings of the Detail-Listing-of-Exception-Transactions report are:

- **SEQ NO.**
  - Sequence in which this transaction entered sync point processing. Seven print positions are provided for this column; therefore, if there are more than 99999999 transactions during the specified analysis period, the sequence number wraps to 0.

- **TRANCODE OR PSB**
  - The transaction code, or PSB name.

- **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, for this case, the following columns are blank. The meaning of nonblank codes A through U is as follows:

---

Chapter 13. Fast Path Log Analysis Utility (DBFULTA0)
Exception Transactions Report

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.

Figure 77 on page 300 is an example of the Summary-of-Exception-Detail-by-Transaction-Code report.
Exception Detail by Transaction Code Report

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.  

**LENG OF INPUT**
The average and maximum values of input message length.  

**LENG OF OUTPUT**
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.

---


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. Figure 78 is an example of the Overall Summary of Transit Times by Transaction Code for IFP Regions.

---

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.
Figure 79 is an example of the Overall Summary of Resource Usage and Contentions for All Transaction Codes and PSBs.

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
All Transaction Codes and PSBs Report

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX</td>
<td>The maximum number of calls per processing interval</td>
</tr>
<tr>
<td>WTS</td>
<td>The total number of times a transaction waited for a buffer to become available</td>
</tr>
<tr>
<td>STL</td>
<td>The total number of times buffer stealing was invoked for the transaction</td>
</tr>
<tr>
<td>TOTL SYNC FAIL</td>
<td>The total number of occurrences of this transaction code that failed synchronization point processing.</td>
</tr>
</tbody>
</table>

CONTENTIONS

The number of control interval contentions

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOT UOW</td>
<td>The total number of times unit-of-work contentions occurred for this transaction code</td>
</tr>
<tr>
<td>TOT OBA</td>
<td>The total number of times overflow buffer area contentions occurred for this transaction code</td>
</tr>
<tr>
<td>CI/SEC</td>
<td>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</td>
</tr>
</tbody>
</table>

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

Related Reading: For more information on the LGNR specification, see [IMS Version 8: Installation Volume 2: System Definition and Tailoring](#).

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 of time. If the time interval is less than one second, then it will default 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. [Figure 80 on page 303](#) is an example of this 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. Figure 81 is an example of this report.

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.

**Summary-of-VSO-Activity Report**

The column headings of the Summary-of-VSO-Activity report are:

- **SHR(0/1)**
- **VSO GETS**
- **VSO PUTS**
- **DASD GETS**
- **DASD PUTS**
- **I/O SCHED**

**Figure 80. Sample Summary of Region Occupancy (Percent) for IFP Regions by PST**

**Figure 81. Sample Summary of VSO Activity**

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.
Summary of VSO Activity Report

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

Figure 82 is an example of the Recapitulation of the Analysis.

(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

Figure 82. Sample Recapitulation of the Analysis

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.
Recapitulation of the Analysis Report

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.

Related Reading: Refer to [IMS Version 8: Installation Volume 2: System Definition and Tailoring](https://example.com) for an explanation of the LGNR specification.

Line (7)  
Total number of times the entire CI was logged for all transaction codes.

JCL Requirements for DBFUTLA0

EXEC

Executes the Fast Path Log Analysis utility.

//EXEC PGM=DBFUTLA0

DD Statements

STEPLIB DD

Describes the program library that contains the DBFUTLA0 load module.
JCL Requirements

//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. This must be the log file from IMS V5.
//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 for DBFULTA0

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:

\[ \text{STARTDAY}=YYDDD \]

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.

**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:

\[ \text{ENDDAY}=YYDDD \]

(last two digits of the year and the sequential number of the day, running from 1 to 366)

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.
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:

\[ \text{START}=HH:MM:SS\{[+|-]HH:MM\} \]

(in hours, minutes, and seconds for a 24-hour clock). 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:

\[ \text{END}=HH:MM:SS\{[+|-]HH:MM\} \]

(in hours, minutes, and seconds for a 24-hour clock). 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 section. 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:

\[ \text{TT (TRANCODE)}=SS.T \]

(in seconds and tenths of seconds)
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

NOT-MESSAGE means transactions are not IFPs.

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 type of information collected about buffer use is as follows:

- 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)
Utility Control Statements

- 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 '5937' 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 type of information collected about data space use is as follows:
- 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 '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. Figure 83 is an example of parameter statements read from the SYSIN data set and of how they are listed in the SYSPRINT data set. 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. [Figure 84 on page 311] is an example of the parameter display. Date information is obtained from the log buffer control record ('42').

SPECIFIED INPUT PARAMETERS:

<table>
<thead>
<tr>
<th>TRANSACTION CODE</th>
<th>EXCEPTION VALUE IN SEC.</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>GLOBAL</em></td>
<td>0.0</td>
</tr>
</tbody>
</table>

Figure 83. Specified Input Parameters
Error Processing for DBFULTA0

User abend codes are not generated.

The following return codes are produced:

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion of analysis</td>
</tr>
<tr>
<td>4</td>
<td>Analysis prematurely ended, partial results produced</td>
</tr>
<tr>
<td>8</td>
<td>Unable to perform analysis</td>
</tr>
<tr>
<td>12</td>
<td>Unable to open DDname SYSPRINT</td>
</tr>
</tbody>
</table>

Related Reading: See [IMS Version 8: Messages and Codes, Volume 1](#) for an explanation of the messages generated by this utility.
Chapter 14. 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).

Related Reading: See MVS/ESA™ Interactive Problem Control System User’s Guide and Reference for information about 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.

Related Reading: Refer to IMS Version 8: Diagnosis Guide and Reference for information about using the Offline Dump Formatter utility to solve problems.

In this Chapter:
- "Interactive Dump Formatter"
- "Migration Considerations" on page 314
- "Restrictions for DFSOFMD0" on page 314
- "Environments for DFSOFMD0" on page 314
- "Input and Output for DFSOFMD0" on page 315
- "IPCS Execution" on page 315

Interactive Dump Formatter

IPCS uses menus on the screen to run the Interactive Dump Formatter. These menus allow you to specify the information to be contained in the dump. The Interactive 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.

Using the Interactive Dump Formatter gives you a menu-driven way to run the Offline Dump Formatter utility without complicated editing of the DFSFRMAT file.

Related Reading: Refer to IMS Version 8: Diagnosis Guide and Reference for a full description of the Interactive Dump Formatter and the IPCS menus.
Migration Considerations

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:

<table>
<thead>
<tr>
<th>Alias</th>
<th>Load Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFSOF810</td>
<td>DFSOFMD0</td>
</tr>
<tr>
<td>DFSAB810</td>
<td>DFSABND0</td>
</tr>
</tbody>
</table>

The IPCS TASKLIB concatenation can contain multiple execution libraries from IMS Version 4, and one execution library from earlier IMS releases.

Restrictions for DFSOFMD0

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 MVS 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 an MVS that is the same level as the MVS 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 MVS system that was dumped.
- 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.
- 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.

Environments for DFSOFMD0

The following sections explain how to use the Offline Dump utility in an IMS online or IMS batch.

IMSI Online Environments

To use the Offline Dump Formatter utility in IMS DB/DC, DCCTL, or DBCTL environments, specify the IMS start parameter option FMTO=D.
Related Reading: For more information on the FMTO= parameter see, IMS Version 8: Installation Volume 2: System Definition and Tailoring.

You can also use a SYSMDUMP DD statement.

**IMS Batch Environments**

To format IMS batch job dumps offline in DBCTL, DB/DC, or DCCTL batch environments, you can request an MVS SYSMDUMP. MVS creates a dump 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.

Related Reading: Refer to the IMSBATCH procedure in IMS Version 8: Installation Volume 2: System Definition and Tailoring for more information on IMS batch job dumps.

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.

**Input and Output for DFSOFMD0**

This utility requires the following input:

- An acceptable machine-readable dump, such as:
  - SDUMP
  - SYSMDUMP
  - Stand-alone dump
  - Dump requested by the MVS 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 the 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. The dump formatter cannot determine the release levels for the concatenated program libraries, but continues under the presumption that they are correctly concatenated.

**IPCS Execution**

To use the Offline Dump Formatter utility under IPCS, you must provide an IMS user control statement.

**Example:** Some examples of the IMS user control statement include:

VERBX DFSOFMD0 'jjjjjjjj[,R][,D]' options

VERBX DFSOFMD0 'jjjjjjjj[,R][,H],FMTIMS(ALL)' options

VERBX DFSOF320 'jjjjjjjj,FMTIMS(SCD)' options
VERBX DFSOF320 'jjjjjjjjjj[R][,N],FMTIMS(AUTO,MIN)' options
VERBX IMSDUMP 'jjjjjjjjjj[R][,D],FMTIMS(SAVEAREA,DISP)' options
VERBX IMSDUMP 'jjjjjjjjjj[R][,D]' 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.

**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.

**Example:** The following is an example of a TSO ALLOCATE command to provide IMS dump format control data set information:

```
ALLOC FI(DFSFRMAT) SHR DA('dump.controldsnname')
```

**Related Reading:** See *MVS/ESA Interactive Problem Control System User's Guide and Reference* for more information about IPCS.

**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
- AUTO, or AUTO,MIN, or AUTO,SUM
- CBT
- DB or DB,MIN
- DBRC
- DC or DC,MIN
- DEDB or DEDB,MIN
- DISPATCH or DISPATCH,MIN
- EMH or EMH,MIN
- LOG or LOG,MIN
- LUM
- MSDB or MSDB,MIN
- QM or QM,MIN
- RESTART
- SAVEAREA, or SAVEAREA,MIN or SAVEAREA,SUM
- SB or SB,MIN
- SCD or SCD,MIN
- 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
IPCS Execution

- SYSPST,system pst address or SYSPST,system pst name
- TRACE,name or TRACE,name,MIN

**Related Reading:** Refer to *IMS Version 8: Diagnosis Guide and Reference* for detailed information about the data areas formatted by these subset options.
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
- Message processing program (MPP)
- Dependent region
- Priority
- Class of the transaction

Any nonrecoverable and 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 MPP is terminated
- 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.

In this Chapter:

- "Restrictions for DFSILTA0"
- "Input and Output for DFSILTA0" on page 320
- "JCL Requirements for DFSILTA0" on page 320


Restrictions for DFSILTA0

The Log Transaction Analysis utility has the following restrictions:

- Log data sets from a batch region are not used.
- Any nonrecoverable and canceled messages are not used.
Log Transaction Analysis

- You must run the Log Merge utility (DFSLTMG0) before you run the Log Transaction Analysis utility against two or more IMS system logs. (The system ID field reflects the order of input to DFSLTMG0.)
- 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.
- Examine control statements for the sort program to determine whether they must be changed, because provision for 256 dependent regions increases the length of the dependent region ID field for the IMS Log-Analysis Report.
- 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/ESA as the utility release level.
- Neither BMPs or messages processed by BMPs are processed.

Input and Output for DFSILTA0

There are three types of input to DFSILTA0:
- IMS log data set. This is required.
- Report title statement. This provides descriptive information for the optional title data set.
- Parameters. There are two optional keyword parameters: ST= and OUT=. These specify what portion of the log data set is to be examined for transactions, and what outputs are to be produced. Parameters can be specified in any combination and should be separated by commas.

DFSILTA0 produces the following output:
- A new IMS log data set, if requested
- A detailed report in input sequence (if NOREPORT is not specified)
- A report, on disk, that can be sorted to produce a sequenced report
- A heading report (if NOREPORT 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 Requirements for DFSILTA0

EXEC

Executes the Log Transaction Analysis utility, DFSILTA0.

Example: This example produces a report but no log data set.

```
//STEP0 EXEC PGM=DFSILTA0,PARM='ST=(hhmmss+HHMM,,mm),
//OUT=NOLOG'
```

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:
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.

**Note:** 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 MVS 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.

**OUT**=

Specifies the desired output. If the OUT= keyword is not specified, the DFSILTA0 defaults produce both a log data set and a report from the log.

**NOLOG**

Specifies that a new IMS log data set is not to be produced.
JCL Requirements

**NOREPORT**
Specifies that no report is to be produced.

**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 heading output data set.

```
//HEADING DD SYSOUT=A
```

**PRINTER DD**
Describes the printed report output data set.

```
//PRINTER DD SYSOUT=A
```

**REPORT DD**
Describes the report output data set. This data set can pass to a sort step. Report entry headings and any checkpoint records are not included in this data set.

```
//REPORT DD DSNAME=&REPORT,DISP=(,PASS),UNIT=SYSDA,
// SPACE=(CYL,(1,1))
```

**LOGIN DD**
Describes the input log data set.

```
//LOGIN DD DSNAME=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 DSNAME=IMS.LOGOUT,DISP=(,PASS),
// VOL=SER=XXXXXX,UNIT=TAPE,DCB=(RECFM=VB,
// LRECL=6004,BKSIZE=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 DSNAME=&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

SORTWK01-12DD
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
Chapter 16. Statistical Analysis Utility (DFSISTS0)

Use the Statistical Analysis utility for analyzing the information on any of the IMS system logs, except those from a batch region. The program modules of this utility reside in IMS.SDFSRESL. The utility consists of modules DFSISTS0, DFSIST20, DFSIST30, and DFSIST40. These modules must be run in sequence.

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.

Related Reading:

In this Chapter:
- “Restrictions for DFSISTS0”
- “Input and Output for DFSISTS0”
- “Examples of DFSISTS0” on page 331
- “JCL Requirements for DFSISTS0” on page 335
- “Utility Control Statements for DFSISTS0” on page 340

Restrictions for DFSISTS0

The Statistical Analysis utility cannot use Common Queue Server (CQS) logs as input because CQS log records have a different format from IMS log records.

Input and Output for DFSISTS0

The following process occurs when you run the Statistical Analysis utility:

1. The selected log data set is passed to module DFSISTS0 (SORT and EDIT PASS1), which edits and sorts the log data set and outputs a modified log data set.
2. The utility sorts the modified log data set (SORT2).
3. The modified log data set is passed to module DFSIST20 (EDIT PASS2), which edits and outputs a new modified log data set.
4. Optional: Either before or after SORT3, the modified log data set can be sent to module DFSIST40 (Message Select and Copy or List), which produces a message list or message data set.
5. The utility sorts the modified log data set (SORT3).
6. The modified log data set is passed to module DFSIST30 (Report Writer), which produces the final statistical reports.

This flow of information is illustrated in Figure 85 on page 326.
Log Records

The log records used by the IMS Statistical Analysis utility are as follows:

01  Input message ready to be put on the destination message queue

03  Output message segment ready to be put on the destination message queue

06  IMS has been started, a date change has occurred, IMS has been stopped, or a FEOV was issued

07  An application program has been terminated

31  The application program issues a "get unique" to retrieve its next message
Message queue manager released a record
Message canceled and a portion of that message has been previously logged
Message has been put on the destination message queue
Message has been taken off the destination message queue

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 02**
This record is created after a command is successfully completed and before the command completion message is sent. If the command is a /LOG or the command must be reprocessed at restart time, a 02 record is written (for example, /ASSIGN). Type 02 log records are not included in the statistics utilities reports, but can be processed by a user-written routine link-edited with DFSIST00. (The information that can be useful to you are the /LOG records.) These /LOG records are entered by either the remote terminal operator or the master terminal operator.

**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 06**
Type 06 records are written when IMS is started (during initialization), when IMS is terminated (immediately prior to closing the log data set), and when a FEOV is issued.

**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 33
The 33 record is written when a message is taken off the input message queue or output message queue.

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=PAGDEDEL (or PAGDEDEL=YES), the message is dequeued when you enter a question mark, PA2 key, or a new input transaction. Option=NPGDEDEL (or PAGDEDEL=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=NPGDEDEL (or PAGDEDEL=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.

SORT and EDIT PASS1 (DFSISTS0)
The functions of SORT and EDIT PASS1 are to:
- Select from the system logs those records used by the statistics programs. (Logs from batch regions do not contain the desired records, and cannot be used.)
- Sort message and queue manager log records so that all segments of a multi-segment message appear together, and enqueue and dequeue records associated with the messages to which they refer.
- Edit the records so that the input message, and all output sent as a result of that input, are contiguous after sorting. Any nonrecoverable and canceled messages are not used.

Concatenation of logs from multiple systems is permitted.

Restrictions:
- The JCL for SORT and EDIT PASS1 must contain a JOBLIB or STEPLIB statement for the library containing the utility program (IMS.SDFSRESL).
- The ',NOTXT' parameter causes the program to ignore the text of the X'01' (input message) and X'03' (output message) log records, thereby reducing the volume
of all sort passes. If you use this parameter, the Message Select and Copy or List utility (DFSIST40) cannot be run. The DFSISDBX suffix is no longer used and will be ignored if you specify it.

**EDIT PASS2 (DFSIST20)**

The function of EDIT PASS2 is to take the records to be used to produce the statistical reports from system messages. If DFSIST40 (Message Select and Copy or List utility) is not run as part of the statistics job stream, approximately 40% of the output of DFSIST20 can be eliminated by coding NOTXT on DFSISTS0 or NOLOG in the SLDS control statement of the Log Archive utility.

**Report Writer (DFSIST30)**

The function of DFSIST30 is to produce the final statistical reports.

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 shows the line and terminal loading by time of day (can be used to determine the line and terminal utilization, peak traffic periods, and so forth).**
  Counts input messages (R), X'01', to IMS from each LTERM, and output messages (S), X'03', to each LTERM from IMS. The report is arranged in line number (relative terminal sequence). A message switch counts as two messages; one from the originating terminal, one to the destination terminal. A broadcast message counts as one message from the originating terminal, and one message each to the destination terminals.

  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. The time for input messages to IMS from each logical terminal is indicated by “R”; the time for output messages to each logical terminal from IMS is indicated by “S”. 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 nth 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 and/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,” are included. The count will be different because of “get unique” issued on which end-of-file is returned.

Task time is set when a request for scheduling is made. The value is the maximum time 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 other than zero in register 15.

- IMS-Accounting Report

Shows start and stop times for the IMS control region.

- Operating Information

- Reports produced, either with or without date control.
- Program determines if input was sorted on date.
- Control break occurs whenever date changes, totals printed, and new report started.
– If not sorted on date, process all the log data sets at one time for a period (such as one week) to produce one summary report.
– The LINECNT=XX parameter can be included on the EXEC statement for the REPORT WRITER (DFSIST30). This is the only parameter expected, and it is optional. If it is not included, the default line count is 36.
– Printing of the different statistical reports is not optional; they are all generated.

Message Select and Copy or List (DFSIST40)

The execution of the Message Select and Copy or List program is optional. You can execute it as a separate step in the same job with the statistical reports, or run it independent of the statistical reports.

This program takes output from the second edit program, DFSIST20, before it is sorted (in line-and-terminal sequence), or after sorting (in transaction-code sequence). Input to this program is specified on the IMSLOGI DD statement, described in “DD Statements” on page 337. To have messages printed in the sequence they occurred (that is, each input message associated with its output message), the input to this program must be &ED34IN.

Examples of DFSISTS0

This section contains examples of the output produced by the Report Writer (DFSIST30) and the Message Select and Copy or List (DFSIST40) programs.

Report Writer (DFSIST30) Output

Following is a list of statistics reports produced by the Report Writer (DFSIST30). Examples of the reports follow. (The report date, which is in the upper right corner of these examples, will not appear unless a sort by date is specified.) The reports produced are:

• Messages—Queued but Not Sent by Destination (Figure 86)
• Messages—Program to Program by Destination (Figure 87 on page 332)
• Line and Terminal (Figure 88 on page 332)
• Messages—Queued but Not Sent by Transaction Code (Figure 89 on page 332)
• Messages—Program to Program by Transaction Code (Figure 90 on page 333)
• Transaction (Figure 91 on page 333)
• Transaction Response (Figure 92 on page 333)
• Application Accounting (Figure 93 on page 334)

<table>
<thead>
<tr>
<th>DESTINATION</th>
<th>TOTAL MESSAGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTRL</td>
<td>1</td>
</tr>
<tr>
<td>PDSW0032</td>
<td>1</td>
</tr>
<tr>
<td>PDSW0043</td>
<td>1</td>
</tr>
<tr>
<td>PDSW0053</td>
<td>1</td>
</tr>
<tr>
<td>PDSW0064</td>
<td>1</td>
</tr>
<tr>
<td>PDSW0082</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 86. Messages—Queued but Not Sent (by Destination)
Examples

**Figure 87. Messages - Program to Program (by Destination)**

<table>
<thead>
<tr>
<th>LINE RTN</th>
<th>TOTAL MESSAGES</th>
<th>TOTAL CHARACTERS</th>
<th>Size 00-07</th>
<th>Hourly Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSW0011</td>
<td>PDSW0011 S</td>
<td>13</td>
<td>2,098</td>
<td>161</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PDSW0012 S</td>
<td>11</td>
<td>352</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PDSW0013 S</td>
<td>14</td>
<td>1,775</td>
<td>126</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PDSW0014 S</td>
<td>15</td>
<td>1,151</td>
<td>76</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PDSW0015 S</td>
<td>12</td>
<td>678</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PDSW0016 S</td>
<td>11</td>
<td>355</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PDSW0017 S</td>
<td>10</td>
<td>351</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PMT01AP S</td>
<td>16</td>
<td>930</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>SYSTEM</td>
<td></td>
<td>S</td>
<td>53,695</td>
<td>5,428,432</td>
</tr>
<tr>
<td>TOTALS</td>
<td></td>
<td>R</td>
<td>23,934</td>
<td>3,367,375</td>
</tr>
</tbody>
</table>

**Notes:**

1. **LINE RTN** = Line Relative Terminal Number
2. **R/S** = Received/Sent

**Figure 88. Line-and-Terminal Report**

<table>
<thead>
<tr>
<th>TRANSACTION CODE</th>
<th>TOTAL MESSAGES</th>
<th>TOTAL MESSAGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>(JMSYS)</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>DE10</td>
<td>39</td>
<td></td>
</tr>
<tr>
<td>DE11</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>DE16</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>DE18</td>
<td>71</td>
<td></td>
</tr>
<tr>
<td>HRQ</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>IT8T</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>OE10</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>OE20</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>OE21</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>OE22</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>OE23</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>OE24</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SC2R</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 89. Messages—Queued but Not Sent (by Transaction Code)**
TRANSACTION REPORT

TRANSACTION CODE

Figures 90, 91, and 92 provide examples of output generated by the DFSISTS0 utility, specifically the Transaction and Response Reports. These reports detail the performance of system programs, including average response times and distribution of responses over various time intervals.

Chapter 16. Statistical Analysis Utility (DFSISTS0) 333
Examples

Figure 93 on page 335 shows an example of the report produced by the Message Select and Copy or List program.
The JCL for execution of the IMS Statistical Analysis utility is given in Figure 95 on page 336. Also see the appropriate operating system Sort/Merge program manual.
JCL Requirements

//STATS JOB 1,NAME,MSGCLASS=A,MSGLEVEL=1,PRTY=8
//JOBLIB DD DSNAME=IMS.SDFSRESL,DISP=SHR
 //*
//ST1 EXEC PGM=DFSIST30
//SORTLIB DD DSNAME=SYS1.SORTLIB,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSDUMP DD SYSOUT=A
//SYSOUT DD SYSOUT=A
//LOGIN DD DSNAME=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,DCB=*,LOGIN
//LOGOUT DD DSNAME=&&EDIT1,DISP=(NEW,PASS),
// UNIT=SYSDA,
// DCB=(RECFM=VB,BLKSIZE=3996,LRECL=3992,BUFNO=3),
// SPACE=(CYL,(5,5))
/*
//ST2 EXEC PGM=SORT,REGION=256K
//SORTLIB DD DSNAME=SYS1.SORTLIB,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SORTIN DD DSNAME=&&EDIT1,DISP=(OLD,DELETE)
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTWK02 DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTWK03 DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTOUT DD DSNAME=&&EDIT1S,DISP=(NEW,PASS),
// UNIT=SYSDA,
// DCB=*,SORTIN,
// SPACE=(CYL,(5,5))
//SYSIN DD *
//SORT FIELDS=(5,1,CH,A,9,4,PD,A,13,38,CH,A),SIZE=E200
/*
/*
//ST3 EXEC PGM=DFSIST20
//EDITDCB1 DD DSNAME=&&EDIT1,DISP=(OLD,DELETE)
//EDITDCB2 DD DSNAME=&&EDIT2,DISP=(NEW,PASS),
// UNIT=SYSDA,
// DCB=(RECFM=VB,BLKSIZE=4016,LRECL=4012,BUFNO=3),
// SPACE=(CYL,(5,5))
//SYSPRINT DD SYSOUT=A
/*
//ST4 EXEC PGM=SORT,REGION=256K
//SORTLIB DD DSNAME=SYS1.SORTLIB,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SORTIN DD DSNAME=&&EDIT2,DISP=(OLD,DELETE)
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTWK02 DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTWK03 DD UNIT=SYSDA,SPACE=(CYL,(5),,CONTIG)
//SORTOUT DD DSNAME=&&ED34IN,DISP=(NEW,PASS),
// UNIT=SYSDA,
// DCB=*,SORTIN,
// SPACE=(CYL,(1,1))
//SYSIN DD *
//SORT FIELDS=(5,1,CH,A,9,4,PD,A,13,40,CH,A),SIZE=E200
/*
//ST5 EXEC PGM=DFSIST30
//EDITDCB2 DD DSNAME=&&ED34IN,DISP=(OLD,PASS)
//PRINTFDCB DD SYSOUT=A,DCB=(BLKSIZE=133,LRECL=133,RECFM=FA)
//SYSPRINT DD SYSOUT=A

Figure 95. JCL for the Statistical Analysis Utility (Part 1 of 2)
The LRECL and BLKSIZE for the log can be calculated as follows:

**LRECL**

(Larger of 1032 or RECLNG operand of the MSGQUEUE macro for IMS.LGMSG) + 16

**BLKSIZE**

Larger of LRECL + 4 or BLKSIZE operand on the IEFRDERDD statement in the IMS cataloged procedure

// EXEC

Does one of the following:

- Invokes the initial selection and sort process in the statistics program jobstream. If DFSIST40 is not run, 'NOTXT' improves performance. Its format is:
  
  //ST1 EXEC PGM=DFSISTS0[PARM=',NOTXT']

- Executes the sort program. You can improve efficiency by providing as much main storage as possible. Sort computer storage size needed based on number of work data sets allocated. Its format is:
  
  //ST2 EXEC PGM=SORT,REGION=256K

- Invokes the second statistics edit module, DFSIST20. If DFSIST40 (Message Select and Edit) is not being run, specify NOTXT on DFSISTS0. Its format is:
  
  //ST3 EXEC PGM=DFSIST20

- Invokes DFSIST30 (Report Writer). Its format is:
  
  //ST5 EXEC PGM=DFSIST30[,PARM='LINECNT=XX']

  If LINECNT is not specified, the default is 36.

- Invokes Message Select and Edit (DFSIST40). This step is optional, and a number of DFSIST0 options apply. This format of the statement is:
  
  //ST6 EXEC PGM=DFSIST40[,PARM='LINECNT=XX']

  If LINECNT is not specified, the default is 36.

### DD Statements

//JOBLIB DD

Describes the program library containing the utility programs. Its format is:

//JOBLIB DD DSNAME=IMS.SDFSRESL,DISP=SHR

//LOGIN DD

Describes the input log data set. Multiple volumes and data sets can be concatenated within one statistics program run. Its format is:

//LOGIN DD DSNAME=IMSLOG,DISP=OLD, VOL=SER=XXXXXX, UNIT=TAPE
JCL Requirements

where XXXXX is the volume serial number of the log data set being processed. (For this example, LRECL=3964 and BLKSIZE=3968.)

//SORTWK01-32 DD
Describes the sort program's work data sets. The space defined can vary. The number of data sets must be at 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 one of the following:

- 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
- Describes the output data set to the sort. Its format is:
  //SORTOUT DD DSNAME=&&EDIT1S,DISP=(NEW,PASS),
  //UNIT=SYSDA,SPACE=(CYL,(5,5)),DCB=*.SORTIN

BLKSIZE and LRECL are the same as for 339.
- Describes the output of the sort that serves as input to DFSIST30 or DFSIST40. Its format is:
  //SORTOUT DD DSNAME=&&ED34IN,UNIT=SYSDA;
  DISP=(NEW,PASS),SPACE=(CYL,(1,1)),
  DCB=*.SORTIN

The normal sort control statement is shown as follows.
SORT FIELDS=(5,1,CH,A,13,40,CH,A),SIZE=XXXX
To sort on date and produce reports under date control, the statement is as follows:
SORT FIELDS=(5,1,CH,A,9,4,PD,A,13,38,CH,A),SIZE=XXXX

//LOGOUT DD
Describes the output of DFSISTS0. This data set is normally a temporary one. It serves as input to the next step (a sort). To break the statistics program into multiple jobs, you must modify this statement. Its normal format is:
//LOGOUT DD DSNAME=&&EDIT1,DISP=(NEW,PASS),
  //UNIT=SYSDA,SPACE=(CYL,(5,5)),DCB=(RECFM=VB,
  //BLKSIZE=3996,LRECL=3992,BUFNO=3)

BLKSIZE and LRECL can be changed here and in subsequent steps. LRECL must be at least as large as LRECL for 337 plus 28 bytes for additional prefix information. Space is a function of the input volume.

//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.

//SYSOUT DD
Describes the message output data set. Its format is:
//SYSOUT DD SYSOUT=A
//SORTIN DD
Does one of the following:
- Describes the input data set to the sort. It is the data set described by the DD statement LOGOUT in the previous step. Its format is:
  //SORTIN DD DSNAME=&&EDIT1,DISP=(OLD,DELETE)
- Refers to the output of DFSIST20. Its format is:
  //SORTIN DD DSNAME=&&EDIT2,DISP=(OLD,DELETE)

//SYSIN DD
Does one of the following:
- Describes the sort's control data set normally in the input stream. It is normally a DD * statement (followed by the sort control statement.)
  Sample sort control statement:
  SORT FIELDS=(5,1,CH,A,9,4,P,F,A,13,36,CH,A),SIZE=XXXX
- Describes the control data set for DFSIST40. It is normally a DD * statement. See "Transaction Code Control Statement" on page 340 for the control statement format.

//EDITDCB1 DD
Describes the input data set to DFSIST20 (output of sort in previous step). Its format is:
//EDITDCB1 DD DSNAME=&&EDIT1S,DISP=(OLD,DELETE)

//EDITDCB2 DD
Does one of the following:
- Describes the output of DFSIST20. Its format is:
  //EDITDCB2 DD DSNAME=&&EDIT2,DISP=(NEW,PASS),
  // UNIT=SYSDA,SPACE=(CYL,(5,5)),
  // DCB=(RECFM=VB,BLKSIZE=4016,LRECL=4012,
  // BUFNO=3)
  Requirement: LRECL must be 20 bytes larger than SORTOUT in the previous step. BLKSIZE must be increased proportionately.
- Describes the input to the report writer (the output of the previous sort). Its format is:
  //EDITDCB2 DD DSNAME=&&ED34IN,DISP=(OLD,PASS)

//PRINTDCB DD
Describes the output of the report writer, normally the output stream. It can be blocked or unblocked, because I/O is performed using QSAM, with QSAM acquiring the buffers. Its format is:
//PRINTDCB DD SYSOUT=A,DCB=(BLKSIZE=133,
  // LRECL=133,RECFM=FA)

//IMSLOGI DD
Describes the input to Message Select and Copy or List (DFSIST40). This program uses the same data set for input as DFSIST30 (that is, output of the second sort) or the output of Edit Pass 2 directly. The format of this statement (assuming use of sorted input) is:
//IMSLOGI DD DSNAME=&&ED34IN,DISP=(OLD,DELETE)

//IMSLOGP DD
Describes the program's output. The format of the statement is the same as the PRINTDCB statement for DFSIST30.

//IMSLOGO DD
Is optional. It can be used to create a data set composed of your messages. Its format is:
Utility Control Statements for DFSISTS0

Message Select and Copy or List 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 of 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 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),(TERM*,I),(TERMINV,,O,ALL)
SYM NAME=(TERMPAY,I,O,TERMA)
SYM NAME=(ALL,,O,TERMA)

- The first parameter is a symbolic terminal name of from 1 to 8 characters.
- The second parameter, I, selects input from this terminal.
- The third parameter, O, selects output generated by input from this terminal.
- You can further qualify the output parameter, O, with another symbolic terminal name. If you do this, only output to that symbolic name which resulted from inputs from the preceding name will be selected. If you specify ALL, all output from the terminal represented by the preceding name is selected.

Hardware Terminal Address Control Statement

The format of the hardware terminal address control statement is:
TERM ADDR=(3,1,I,O),(42,3,,O,21,A),(1,ALL,I,O)

- Selection by hardware terminal name is similar to selection by terminal symbolic name, except that, instead of symbolic name, the line number and relative terminal number are specified.
- The first parameter is the line number or ALL.
- The second parameter is the relative terminal number on the line or ALL.
- The third and fourth parameters are I and O for selection of input to and output from this terminal.
- Output can be further qualified (similar to symbolic terminal output).
- If an output message is queued but not sent, it is not selected, even if ALL is specified. The SYM NAME control statement must be used.

VTAM Terminal Name Control Statement

The format of the VTAM terminal name control statement is:
VTERM NAME=(L3270A,I,O),(L3270B,,O,L3270C)

- Selection by VTAM terminal name is similar to selection by terminal symbolic name, except that, instead of symbolic name, the node names are specified.
- The first parameter is the node name or ALL.
- The second parameter, I, selects input from this terminal.
- The third parameter, O, selects output generated by input from this terminal.
- Output can be further qualified (similar to symbolic terminal output). If you specify ALL, all output from the terminal represented by the preceding name is selected.
- If an output message is queued but not sent, it is not selected, even if ALL is specified. The SYM NAME control statement must be used.

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:
JCL Requirements

- 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.
Part 5. Interpreting IMS Reports

Chapter 17. Interpreting IMS Monitor Reports .......................... 347
  Transaction Flow and IMS Monitor Events .................................. 347
  IMS Monitor Trace Event Intervals ........................................... 350
  Overview of IMS Monitor Reports ............................................ 351
    Sequence of Report Output .................................................. 351
    Units of Measure in IMS Monitor Reports ................................ 352
  Documenting the Monitoring Run ............................................ 352
    Adding to the System-Configuration Report Data ......................... 352
    Recording the Monitor Trace Interval .................................... 352
    Completing the Monitor Run Profile ...................................... 353
    Verifying IMS Monitor Report Occurrences ................................ 354
  Monitoring Activity in Dependent Regions ................................ 355
    Detecting Database Processing Intent Conflicts .......................... 358
    Examining the Effects of Checkpoints .................................... 359
    Measuring Region Occupancy ............................................... 359
  Monitoring Application Program Elapsed Time ............................ 359
  Monitoring I/O for Application Program DL/I Calls ...................... 362
  Monitoring MFS Activity ...................................................... 366
  Monitoring Message Queue Handling ....................................... 367
    Detecting Checkpoint Effects ............................................. 368
    Transaction Queueing Report .............................................. 368
  Monitoring Database Buffers ................................................ 369
  Monitoring Line Activity ...................................................... 371
  Monitoring Message Handling Efficiency .................................. 372
  IMS Internal Resource Usage ................................................ 372
    Pool Space Failure .......................................................... 372
    Programs Experiencing Deadlock ........................................... 372
    IMS Latch Conflict .......................................................... 373
  Using Frequency Distributions from IMS Monitor Output ................. 375
    How to Get a Frequency Distribution Output .............................. 375
    How Frequency Distribution Ranges Are Defined ......................... 377
    Interpreting Distribution Appendix ...................................... 379
  Interpreting IMS Monitor MSC Reports ..................................... 380
    Determining Cross-System Queueing ...................................... 380
    Assessing the Effect of Link Loading .................................... 381
    Assessing Link Queueing Times ............................................ 382
  Extracting Multiple System Transaction Statistics ...................... 383
    Controlling the Log Merge ................................................ 383
    Interpreting the Transaction Analysis Report ............................ 383

Chapter 18. Interpreting IMS Monitor Reports for DBCTL .................. 385
  IMS Monitor Trace Event Intervals ........................................... 386
  Overview of IMS Monitor Reports ............................................ 387
    Sequence of Report Output .................................................. 387
    Units of Measure in IMS Monitor Reports ................................ 388
  Documenting the Monitoring Run ............................................ 388
    Adding to the System Configuration Report Data ......................... 388
    Recording the Monitor Trace Interval .................................... 388
    Completing the Monitor Run Profile ...................................... 389
    Verifying IMS Monitor Report Occurrences ................................ 390
  Monitoring Activity in Dependent Regions ................................ 390
    Detecting Database Processing Intent Conflicts .......................... 394
    Examining the Effects of Checkpoints .................................... 395
Chapter 21. Interpreting Statistical-Analysis and Log-Transaction Reports

Statistical Analysis Utility Reports

Calculating Transaction Loads
Assessing Program-to-Program Traffic
Obtaining Counts of Unsent Messages

Auditing Critical Transactions

Log Transaction Analysis Utility Reports

Examining Scheduling Activity

IMS Accounting Information

Using the Application-Accounting Report

Using IMS Transaction Profiles
This chapter describes:

- The events that the IMS Monitor collects
- The content of the reports produced by the IMS Monitor Report Print Program

The IMS Monitor output reflects the IMS DB/DC environment as a whole. A subset of the reports deals with database calls and buffering. Interpreting this data for batch application performance and to verify database design are considered separate tasks allied with database administration.

Related Reading: For more information on these reports, see [IMS Version 8: Utilities Reference: Database and Transaction Manager](#).

In this Chapter:

- “Transaction Flow and IMS Monitor Events”
- “IMS Monitor Trace Event Intervals” on page 350
- “Overview of IMS Monitor Reports” on page 351
- “Documenting the Monitoring Run” on page 352
- “Monitoring Activity in Dependent Regions” on page 355
- “Monitoring Application Program Elapsed Time” on page 359
- “Monitoring I/O for Application Program DL/I Calls” on page 362
- “Monitoring MFS Activity” on page 366
- “Monitoring Message Queue Handling” on page 367
- “Monitoring Database Buffers” on page 369
- “Monitoring Line Activity” on page 371
- “Monitoring Message Handling Efficiency” on page 372
- “IMS Internal Resource Usage” on page 372
- “Using Frequency Distributions from IMS Monitor Output” on page 375
- “Interpreting IMS Monitor MSC Reports” on page 380
- “Extracting Multiple System Transaction Statistics” on page 383

Transaction Flow and IMS Monitor Events

For an overall picture of the events, system activities, and usage of storage areas (buffer pool or data set) for which the IMS Monitor gathers timings, see Table 21. The leftmost column shows, from top to bottom, a sequence of processing events; each event is related to IMS Monitor reported items in the notes that follow the table.

<table>
<thead>
<tr>
<th>Flow</th>
<th>Event</th>
<th>Activity</th>
<th>Pool</th>
<th>Data Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Wait for poll</td>
<td>Waiting</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>2</td>
<td>Data Transfer</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>3</td>
<td>Input message processing</td>
<td>Device CIOP MFS</td>
<td>QBUF</td>
<td>Message queue</td>
</tr>
<tr>
<td>4</td>
<td>Input queuing</td>
<td>Waiting</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
### Table 21. Transaction Flow and IMS Monitor Events Description (continued)

<table>
<thead>
<tr>
<th>Flow</th>
<th>Event</th>
<th>Activity</th>
<th>Pool</th>
<th>Data Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Scheduling</td>
<td>Scheduling</td>
<td>QBUF</td>
<td>Message queue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PSB load</td>
<td>PSB, PSBW, DPSB</td>
<td>IMS.ACBLIBx</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DMB load</td>
<td>DMBP</td>
<td>IMS.ACBLIBx</td>
</tr>
<tr>
<td>6</td>
<td>Program load</td>
<td>Program load</td>
<td></td>
<td>IMS.PGMLIB</td>
</tr>
<tr>
<td>7</td>
<td>Program initialization</td>
<td>Initializing</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>8</td>
<td>Message queue GU</td>
<td>Primed GU call</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>9</td>
<td>Program Execution</td>
<td>DC calls</td>
<td>QBUF</td>
<td>Message queue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DL/I Elapsed</td>
<td>DB calls</td>
<td>OSAM/VSAM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wait Elapsed</td>
<td>DB I/Os</td>
<td>OSAM/VSAM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SPA insert</td>
<td>CWAP</td>
<td>IMS.SPA</td>
</tr>
<tr>
<td>10</td>
<td>Output message insert</td>
<td>DC ISRT call</td>
<td>QBUF</td>
<td>Message queue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DC GU call</td>
<td>QBUF</td>
<td>Message queue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sync point</td>
<td>OSAM/VSAM</td>
<td>Databases</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(MODE=SNGL)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Wait for sync point</td>
<td>(Only for MODE=MULT)</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>12</td>
<td>Program termination</td>
<td>Program termination</td>
<td>OSAM/VSAM</td>
<td>Databases</td>
</tr>
<tr>
<td>13</td>
<td>Wait for selection</td>
<td>Waiting</td>
<td>QBUF</td>
<td>Message Queue</td>
</tr>
<tr>
<td>14</td>
<td>Output message processing</td>
<td>Message send</td>
<td>QBUF</td>
<td>Message queue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MFS</td>
<td>MFP</td>
<td>IMS.FORMATx</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Device</td>
<td>CIOP</td>
<td>(For lines)</td>
</tr>
<tr>
<td>15</td>
<td>Data transfer</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>16</td>
<td>Output queue processing</td>
<td>Dequeuing</td>
<td>QBUF</td>
<td>Message queue</td>
</tr>
</tbody>
</table>

**Notes to Table 21:**

1. The time waiting for poll is not recorded by the IMS Monitor.
2. Line activity in terms of data transmitted and IMS communication sub-task activity are recorded for all messages. Input message activity is not separated from output message activity.
3. During input message processing, the device dependent processing and the use of the communication I/O pool (CIOP) are recorded. If message format service (MFS) is required, the use of the message format buffer pool and any I/O to the active IMS.FORMATA/B library are recorded. The input message is then placed in a message queue buffer (QBUF) with possible I/O to the message queue data sets. The IMS Monitor reporting does not distinguish this activity from output message processing.
4. Time spent waiting on the input queue is not recorded by the IMS Monitor.
5. IMS schedules the processing program into a region and, as part of this action, accesses the message queue or the QBUF pool so that it can present the message to the program. At this time, the required PSB and physical database blocks are made available in the PSB pool and the DMB pool. If they are not
already in the pool, they are retrieved from the active IMS.ACBLIBA/B data set. These events are summarized under SCHEDULING AND TERMINATION in IMS Monitor reports.

6. Next, the application program is loaded into the region from IMS.PGMLIB, or initialized if already resident in the region. This processing is included as part of SCHEDULE TO FIRST CALL in IMS Monitor reports.

7. The initialization performed by the application program is included in this period of time, up to the time of the first message queue or database call. This processing is considered part of the scheduling process because it is not repeated when multiple transactions are processed for one scheduling event. This processing is recorded as part of the SCHEDULE TO FIRST CALL in IMS Monitor reports.

8. The event of performing the first DL/I GU call to obtain the first message segment is not separately recorded. The processing to prime the application with this message is included with the SCHEDULE TO FIRST CALL.

9. The program elapsed event is measured from the first call to termination of the processing program. The total elapsed time is recorded in IMS Monitor reports as ELAPSED EXECUTION.

- Each DL/I call, whether DC or DB, is individually recorded along with its use of message queues or database data sets and their respective pools. Each external subsystem call is individually recorded. These events are recorded under CALLS in IMS Monitor reports.

- When a DL/I call causes I/O activity, the time spent waiting for the data is recorded as WAIT time and as the number of I/Os. When a program’s database processing intent and other update activity are in contention, this also contributes to WAIT time. For each external subsystem call, the time spent in external subsystem processing is recorded separately as WAIT time. When processing is suspended because of intent, the elapsed time is recorded in IDLE FOR INTENT IMS Monitor report items. When processing is suspended because the region is designated as wait-for-input and no input message is available, the time spent waiting for the next input message is excluded from elapsed time intervals and wait times. Wait-for-input time appears only in **WFI items on the program I/O report.

- If the message processing is part of a conversational transaction, the activity in the communications work area pool (CWAP) and the IMS.SPA data set is recorded.

10. The DL/I ISRT call for the response to the message just processed is recorded for the transaction.

As processing continues, there are synchronization points such as a GU to the message queue for another input message or a checkpoint call. The database and message queue I/O to commit the program processing is recorded. Checkpointing by IMS or by the processing program is recorded separately.

11. If the processing is for multiple messages before synchronization (MODE=MULT), there can be a wait for sync point. This time is not recorded by the IMS Monitor.

12. The processing after exiting from the application program, the program termination events, is included with the initial scheduling time. It is part of SCHEDULING AND TERMINATION.

13. The output message waits for the selection to be transmitted to the terminal. The duration of any wait time on the output queue is not recorded by the IMS Monitor.

14. After the exit from the application program and termination, output message processing events are recorded. These include message format and device
Transaction Flow and IMS Monitor Events

dependent processing as well as sending the output message. The IMS Monitor reporting does not distinguish this activity from input message processing.

15. Line activity in terms of data transmitted and IMS communication sub-task activity are recorded for all messages. Output message activity is not separated from input message activity.

16. The processing to remove the output message from the queue is recorded.

IMS Monitor Trace Event Intervals

The IMS Monitor trace interval is bounded by the master terminal operator’s use of the /TRACE command between the start and stop command entries. The IMS Monitor can also be stopped by the expiration of any interval specified. The online IMS events are recorded in IMS Monitor records placed in the IMSMON data set. The event timings are related to dependent region activity. Figure 96 on page 351 shows the boundaries of the timed event intervals.

The Monitor trace interval includes the following intervals:

- Scheduling and Termination
  - No Messages
  - Block loader busy
  - Intent failures (exclusive intent and data sharing) and Schedule failures (PSB busy and space failure)
  - Sched/Term elapsed
    - NOT-WAIT
    - ACBLIB waits
    - DB Flush waits
    - DB CLOSE waits
- Region occupancy (which overlaps with all of Sched/Term elapsed)
  - Schedule to first call
  - Elapsed execution

The NOT-WAIT time for a region is the elapsed time not accounted for by wait time. Any delay coming from either paging or the processor being dispatched for a higher priority task results in an increase in the NOT-WAIT times.
Overview of IMS Monitor Reports

A list of reports available from data collected by the IMS Monitor, together with the principal performance data they contain, is shown in Table 22.

**Related Reading:** For those reports marked by a “DB”, see *IMS Version 8: Utilities Reference: Database and Transaction Manager*.

The reports marked “MSC” in the list are only produced when MSC is active. The MSC reports and their interpretation are in “Interpreting IMS Monitor MSC Reports” on page 380.

### Sequence of Report Output

The order of the reports listed in Table 22 matches the sequence of the output from the IMS Monitor Report Print Program. The duration or constraints of a monitoring snapshot might not include certain events necessary for an individual report, in which case only report headings or partial data are produced.

<table>
<thead>
<tr>
<th>Report Name</th>
<th>Principal Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Configuration</td>
<td>Monitor run documentation</td>
</tr>
<tr>
<td>Message Queue Pool</td>
<td>Buffering and message I/O per transaction</td>
</tr>
<tr>
<td>Database Buffer Pool (DB)</td>
<td>Count of DB calls and I/O per transaction</td>
</tr>
<tr>
<td>VSAM Buffer Pool (DB)</td>
<td>Count of inserts and I/Os</td>
</tr>
<tr>
<td>Message Format Buffer Pool</td>
<td>Count of blocks fetched and I/Os</td>
</tr>
<tr>
<td>Latch Conflict Statistics</td>
<td>IMS internal processing</td>
</tr>
<tr>
<td>General Wait Time Events</td>
<td>Wait times for SNAPQ</td>
</tr>
<tr>
<td>Region and Jobname</td>
<td>Monitor run documentation</td>
</tr>
<tr>
<td>Region Summary</td>
<td>Elapsed times and count of DL/I calls</td>
</tr>
<tr>
<td>Region Wait</td>
<td>Wait times</td>
</tr>
<tr>
<td>Programs by Region</td>
<td>Elapsed times for region usage</td>
</tr>
<tr>
<td>Program Summary</td>
<td>Overall program statistics</td>
</tr>
<tr>
<td>Program I/O (DB)</td>
<td>Wait times/PCB</td>
</tr>
</tbody>
</table>
Overview of IMS Monitor Reports

Table 22. IMS Monitor Reports Output Sequence and Information  (continued)

<table>
<thead>
<tr>
<th>Report Name</th>
<th>Principal Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication Summary</td>
<td>Elapsed times for lines</td>
</tr>
<tr>
<td>Communication Wait</td>
<td>Wait times by line</td>
</tr>
<tr>
<td>Line Functions</td>
<td>Count and size of blocks transmitted</td>
</tr>
<tr>
<td>MSC Traffic (MSC)</td>
<td>Count and routing of transactions</td>
</tr>
<tr>
<td>MSC Summaries (MSC)</td>
<td>Count of transactions by destination</td>
</tr>
<tr>
<td>MSC Queuing Summary (MSC)</td>
<td>Count and queuing time by link</td>
</tr>
<tr>
<td>Transaction Queuing</td>
<td>Queue loading statistics</td>
</tr>
<tr>
<td>Reports</td>
<td>Count of space failures and deadlocks</td>
</tr>
<tr>
<td>Run Profile</td>
<td>Monitor run documentation</td>
</tr>
<tr>
<td>Call Summary (DB)</td>
<td>Call counts and timings/segment type</td>
</tr>
<tr>
<td>Distribution Appendix</td>
<td>Event frequency distributions</td>
</tr>
</tbody>
</table>

Units of Measure in IMS Monitor Reports

The majority of the data items in IMS Monitor reports are elapsed times. These are normally expressed in microseconds. An entry of 1876534 represents 1.876534 seconds or 1876 milliseconds. Any times that do not follow this convention show the unit of measure on the report.

You can also find counts of events under the heading OCCURRENCES, and some figures that represent the number of bytes.

Documenting the Monitoring Run

For each trace interval there are several general reports or overall summaries of the processing that took place. You can use these reports as part of your IMS Monitor run documentation.

It is important to record, as accurately as possible the conditions under which the trace was taken. Your documentation can include system status information obtained by the /DISPLAY command several times before and after the trace, an expected profile of the application program activity, and any desired processing events. The trace interval should represent typical processing loads and not be a biased or inadequate historical record.

Adding to the System-Configuration Report Data

The first general report titled SYSTEM CONFIGURATION is found under the page heading BUFFER POOL STATISTICS. It shows the modification level of the IMS and MVS systems. The system configuration output is illustrated in Figure 97 on page 353.

Recording the Monitor Trace Interval

The heading of most IMS Monitor reports carries the trace start and stop times. It is shown in the format YEAR DAY (Julian) HH:MM:SS. The overall length of the trace interval is given in milliseconds under the title MONITOR OVERHEAD DATA. The following line shows how many trace records were placed on the IMSMON data set. An example of the monitor trace interval recording is shown in Figure 97 on page 353.
Completing the Monitor Run Profile

A compact set of processing ratios is found at the end of the Run-Profile report. The statistics summarize, for the monitor interval, the transaction throughput and the degree of DL/I and I/O activity. An example of the report is shown in Figure 98.

The lower part of the Run-Profile report shows several ratios:
- Program elapsed time to DL/I elapsed time for each region
- DL/I elapsed time to wait time during DL/I processing
- Program elapsed time to other subsystem call elapsed time
- DL/I elapsed time to other subsystem call elapsed time

Each dependent region is identified by a sequence number, starting at region 1.

You can match the regions to the name of the MVS job using the Region-and-Jobname report. The job names correspond to the step names on the EXEC statements of all the dependent regions started by the operator before the trace was started. The region job names are included on the monitor output page with the heading GENERAL REPORTS, as illustrated in Figure 109 on page 368.
Some generalized processing ratios are given at the end of several buffer pool statistics reports. You can include them in the documented profile of the trace interval. These are not specific to one application or system resource but can be used as indicators of variation across a series of monitor runs.

The ratios are:

- The total number of OSAM reads + OSAM writes + all waits divided by the total number of transactions.
  From the Message-Queue-Pool report (see Figure 108 on page 367), this ratio indicates on a per transaction basis the physical I/O activity required to handle the message queuing function.

- The total number of OSAM reads + OSAM writes + BISAM reads divided by the total number of transactions.
  From the Database-Buffer-Pool report (see Figure 111 on page 370), this ratio indicates on a per transaction basis the physical I/O activity required to handle the database buffering function.

- The total prefetch I/Os + immediate fetch I/Os + directory I/Os divided by the total number of transactions.
  From the Message Format Buffer Pool report (see Figure 107 on page 367), this ratio indicates on a per transaction basis the physical I/O activity required to handle the MFS function.

### Verifying IMS Monitor Report Occurrences

When you examine the output from the IMS Monitor Report Print program, the presence of a report heading does not necessarily mean that appropriate data is listed. System definition options and utility control statements affect the content of the output as follows:

- The output does not include a Call-Summary report unless a control statement specifies DLI.

- The output does not include a set of Distribution reports unless a control statement specifies DIS or DISTRIBUTION. The column headed DISTRIBUTION NUMBER that occurs on many of the reports contains cross-references to items included in the Distribution reports.

- The output consists of just a Call Summary report if a control statement specifies ONLY DLI.

Because many of the summary reports require system status to calculate the difference between start and end values, and this status is obtained during the \TRACE \SET \OFF processing, the IMS Monitor execution must end before termination of the IMS control region. If the trace was not stopped properly, the following message is issued:

NO QUEUE BUFFER POOL TRACES AT END TIME ON MONITOR LOG TAPE
****QUEUE BUFFER POOL REPORT CANCELLED****

Similarly, other summary reports are not produced.

The series of reports with the title BUFFER POOL STATISTICS do not include a VSAM BUFFER POOL section unless one of the databases in IMS.ACBLIB uses the VSAM access method. If VSAM is not used, the following message is issued:

NO VSAM BUFFER POOL TRACES ON MONITOR LOG TAPE
****VSAM BUFFER POOL REPORT CANCELLED****
The section MESSAGE FORMAT BUFFER POOL is included only if your system definition specifies devices using Message Format Service.

If the source data used to formulate a particular IMS Monitor report, or a section of that report, has not been recorded by the IMS Monitor during the trace interval, the report contains only the headings.

Monitoring Activity in Dependent Regions

The IMS Monitor gathers timing information for every dependent region identified in the /trace command active during the trace interval. It records the total of the elapsed times for each event, the maximum individual time encountered, and the average time.

There are three major reports that display timings. The reports and a list of their content are:

- **Region-Summary Report**
  - Scheduling and termination
  - Schedule end to first call
  - Elapsed execution with separate summaries shown for:
    - DL/I calls
    - External subsystem service and command calls
    - External subsystem database access calls
    - Checkpoint processing
    - Region occupancy

- **Region Wait**
  - Waits during scheduling and termination
  - Waits during DL/I calls
  - Waits during external subsystem calls
  - Waits during checkpoint

- **Programs by Region**
  - Elapsed execution
  - Schedule end to first call

These three reports are illustrated in Figure 99 on page 356, Figure 100 on page 357, and Figure 101 on page 358.

Activities for dependent regions are placed in five categories:

- **Elapsed time for scheduling and termination**
  The scheduling process includes many preparatory events such as block loading from an active IMS.ACBLIBA/B data set, and obtaining ownership of the PSB. The time required to terminate the region activity after the application program ends is also included.

- **Elapsed time from end of schedule to first call**
  This time is reserved for application program initialization and housekeeping prior to an initial call (to the message queue, a database, or an external subsystem) that marks the beginning of control program services. It is a measure of processing that is not repeated when multiple transactions are processed in a single scheduling.

- **Program elapsed time, including all calls**
Monitoring Activity in Dependent Regions

This time encompasses the major application program processing, measured from the first call to the return or exit from the program.

- **Elapsed time performing DL/I calls**
  This time includes all DL/I calls. Each DL/I call event is measured from the time of the call to the return to the application program.

- **Elapsed time performing external subsystem calls**
  This time includes all external subsystem calls. Each external subsystem event is measured from the time of the call to the return to IMS.

<table>
<thead>
<tr>
<th>SCHEDULING AND TERMINATION</th>
<th>OCCURRENCES</th>
<th>TOTAL</th>
<th>ELAPSED</th>
<th>MEAN</th>
<th>MAXIMUM</th>
<th>OCCURRENCES</th>
<th>TOTAL</th>
<th>NOT WAIT</th>
<th>TIME(ELAPSED-WAIT)</th>
<th>MEAN</th>
<th>MAXIMUM</th>
<th>DISTRIBUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>REGION</strong> 3</td>
<td>5</td>
<td>4146</td>
<td>829</td>
<td>948</td>
<td>948</td>
<td>4146</td>
<td>829</td>
<td>948</td>
<td>948</td>
<td></td>
<td></td>
<td>287A,B</td>
</tr>
<tr>
<td><strong>REGION</strong> 6</td>
<td>7</td>
<td>6028</td>
<td>861</td>
<td>1067</td>
<td>1067</td>
<td>6028</td>
<td>861</td>
<td>1067</td>
<td>1067</td>
<td></td>
<td></td>
<td>214A,B</td>
</tr>
<tr>
<td><strong>REGION</strong> 8</td>
<td>8</td>
<td>6847</td>
<td>855</td>
<td>1098</td>
<td>1098</td>
<td>6847</td>
<td>855</td>
<td>1098</td>
<td>1098</td>
<td></td>
<td></td>
<td>129A,B</td>
</tr>
<tr>
<td><strong>REGION</strong> 10</td>
<td>7</td>
<td>9664</td>
<td>1380</td>
<td>3688</td>
<td>3688</td>
<td>9664</td>
<td>1380</td>
<td>3688</td>
<td>3688</td>
<td></td>
<td></td>
<td>272A,B</td>
</tr>
<tr>
<td><strong>REGION</strong> 47</td>
<td>6</td>
<td>5482</td>
<td>913</td>
<td>1021</td>
<td>1021</td>
<td>5482</td>
<td>913</td>
<td>1021</td>
<td>1021</td>
<td></td>
<td></td>
<td>145A,B</td>
</tr>
<tr>
<td><strong>REGION</strong> 49</td>
<td>3</td>
<td>2612</td>
<td>870</td>
<td>917</td>
<td>917</td>
<td>2612</td>
<td>870</td>
<td>917</td>
<td>917</td>
<td></td>
<td></td>
<td>443A,B</td>
</tr>
<tr>
<td><strong>TOTALS</strong></td>
<td>123</td>
<td>126042</td>
<td>1024</td>
<td></td>
<td></td>
<td>126042</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SCHEDULE TO FIRST CALL**

| **REGION** 1                | 1           | 15479797 | 15479797 | 15479797 | 555  |
| **REGION** 2                | 1           | 22376350 | 22376350 | 22376350 | 564  |
| **REGION** 3                | 1           | 15169488 | 15169488 | 15169488 | 578  |
| **REGION** 4                | 1           | 48146258 | 48146258 | 48146258 | 584  |
| **REGION** 48               | 1           | 795351   | 795351   | 795351   | 592  |
| **REGION** 49               | 4           | 2960425  | 740106   | 741746   | 442  |
| **REGION** 50               | 1           | 15713464 | 15713464 | 15713464 | 575  |
| **TOTALS**                 | 168         | 514286738| 3061230  |         |      |

**ELAPSED EXECUTION**

| **REGION** 1                | 1           | 290146255| 290146255| 290146255| 1    |
| **REGION** 2                | 1           | 252290108| 252290108| 252290108| 2    |
| **REGION** 3                | 1           | 259496970| 259496970| 259496970| 3    |
| **REGION** 4                | 1           | 322812716| 322812716| 322812716| 4    |
| **REGION** 48               | 1           | 273871107| 273871107| 273871107| 48   |
| **REGION** 49               | 4           | 271703421| 67925855 | 155176058| 49   |
| **REGION** 50               | 1           | 290379922| 290379922| 290379922| 50   |
| **TOTALS**                 | 173         | 1423854014| 82303700 |        |

**TOTALS**

<table>
<thead>
<tr>
<th>DL/I CALLS</th>
<th>OCCURRENCES</th>
<th>TOTAL</th>
<th>ELAPSED</th>
<th>MEAN</th>
<th>MAXIMUM</th>
<th>DISTRIBUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>REGION</strong> 1</td>
<td>60</td>
<td>264626241</td>
<td>4410437</td>
<td>88981490</td>
<td>263813671</td>
<td>4396894</td>
</tr>
<tr>
<td><strong>REGION</strong> 2</td>
<td>223</td>
<td>230505269</td>
<td>1033655</td>
<td>60487588</td>
<td>27368742</td>
<td>1019590</td>
</tr>
<tr>
<td><strong>REGION</strong> 3</td>
<td>29</td>
<td>257704383</td>
<td>8886358</td>
<td>69005314</td>
<td>257704383</td>
<td>8886358</td>
</tr>
<tr>
<td><strong>REGION</strong> 4</td>
<td>4</td>
<td>313735347</td>
<td>396130</td>
<td>52439653</td>
<td>31060935</td>
<td>392188</td>
</tr>
<tr>
<td><strong>REGION</strong> 49</td>
<td>592</td>
<td>262886317</td>
<td>444064</td>
<td>30202068</td>
<td>234394017</td>
<td>395935</td>
</tr>
<tr>
<td><strong>REGION</strong> 50</td>
<td>36</td>
<td>242591451</td>
<td>6738651</td>
<td>48651260</td>
<td>242591451</td>
<td>6738651</td>
</tr>
<tr>
<td><strong>TOTALS</strong></td>
<td>18632</td>
<td>12388985286</td>
<td>664018</td>
<td>12024562411</td>
<td>645371</td>
<td>12024562411</td>
</tr>
</tbody>
</table>

**FOR INTENT**

<table>
<thead>
<tr>
<th>OCCUPANCY</th>
<th>CHECKPOINT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>REGION</strong> 1</td>
<td>100.0%</td>
</tr>
<tr>
<td><strong>REGION</strong> 2</td>
<td>100.0%</td>
</tr>
<tr>
<td><strong>REGION</strong> 3</td>
<td>100.0%</td>
</tr>
<tr>
<td><strong>REGION</strong> 4</td>
<td>100.0%</td>
</tr>
<tr>
<td><strong>REGION</strong> 48</td>
<td>100.0%</td>
</tr>
<tr>
<td><strong>REGION</strong> 49</td>
<td>100.0%</td>
</tr>
<tr>
<td><strong>REGION</strong> 50</td>
<td>100.0%</td>
</tr>
</tbody>
</table>

**REGION OCCUPANCY**

| **REGION** 1 | 100.0% |
| **REGION** 2 | 100.0% |
| **REGION** 3 | 100.0% |
| **REGION** 4 | 100.0% |
| **REGION** 48 | 100.0% |
| **REGION** 49 | 100.0% |
| **REGION** 50 | 100.0% |

**Figure 99. Region-Summary Report**
### Monitoring Activity in Dependent Regions

**Figure 100. Region-Wait Report**

<table>
<thead>
<tr>
<th>Region</th>
<th>Total</th>
<th>Mean</th>
<th>Maximum</th>
<th>Function</th>
<th>Module</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMS Monitor</td>
<td>181816</td>
<td>16528</td>
<td>24375</td>
<td>DO=IMMSTR2A</td>
<td>DBH</td>
<td>117</td>
</tr>
<tr>
<td>Scheduling + Termination</td>
<td>112831</td>
<td>14103</td>
<td>17846</td>
<td>DO=IMMSTR1A</td>
<td>DBH</td>
<td>118</td>
</tr>
<tr>
<td></td>
<td>85460</td>
<td>17092</td>
<td>3317</td>
<td>DO=IMMSTR3A</td>
<td>DBH</td>
<td>119</td>
</tr>
<tr>
<td></td>
<td>58420</td>
<td>11604</td>
<td>14643</td>
<td>DO=IMINDEXA</td>
<td>VBH</td>
<td>120</td>
</tr>
<tr>
<td></td>
<td>173866</td>
<td>14488</td>
<td>22152</td>
<td>DO=PRDCNTA</td>
<td>VBH</td>
<td>121</td>
</tr>
<tr>
<td></td>
<td>100576</td>
<td>33525</td>
<td>68373</td>
<td>DO=IMMSTR2B</td>
<td>DBH</td>
<td>428</td>
</tr>
<tr>
<td></td>
<td>17921</td>
<td>17921</td>
<td>17921</td>
<td>DO=IMMSTR3B</td>
<td>DBH</td>
<td>429</td>
</tr>
<tr>
<td></td>
<td>17195</td>
<td>17195</td>
<td>17195</td>
<td>DO=IMMSTR1B</td>
<td>DBH</td>
<td>430</td>
</tr>
<tr>
<td></td>
<td>13577</td>
<td>13577</td>
<td>13577</td>
<td>DO=IMINDEXB</td>
<td>VBH</td>
<td>431</td>
</tr>
<tr>
<td></td>
<td>49928</td>
<td>16642</td>
<td>20396</td>
<td>DO=PRDCNTB</td>
<td>VBH</td>
<td>432</td>
</tr>
<tr>
<td></td>
<td>10973</td>
<td>2743</td>
<td>2787</td>
<td>DO=ITEMACTB</td>
<td>DBH</td>
<td>453</td>
</tr>
<tr>
<td></td>
<td>37680</td>
<td>18840</td>
<td>27664</td>
<td>DO=IAINDEXB</td>
<td>VBH</td>
<td>454</td>
</tr>
<tr>
<td></td>
<td>1500067</td>
<td>30613</td>
<td>138284</td>
<td>DO=INVENTRA</td>
<td>DBH</td>
<td>472</td>
</tr>
<tr>
<td></td>
<td>345595</td>
<td>15025</td>
<td>27613</td>
<td>DO=VENDORDA</td>
<td>VBH</td>
<td>473</td>
</tr>
<tr>
<td></td>
<td>34292</td>
<td>34292</td>
<td>34292</td>
<td>P=VENDORDA...1</td>
<td>498</td>
<td></td>
</tr>
<tr>
<td></td>
<td>14612</td>
<td>14612</td>
<td>14612</td>
<td>P=VNSINDXA...1</td>
<td>499</td>
<td></td>
</tr>
<tr>
<td></td>
<td>69203</td>
<td>11533</td>
<td>19492</td>
<td>DO=VNSINDXA</td>
<td>VBH</td>
<td>500</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>136</td>
<td>3132672</td>
<td>23034</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Monitoring Activity in Dependent Regions

<table>
<thead>
<tr>
<th>IMS MONITOR</th>
<th><strong><strong>PROGRAMS BY REGION</strong></strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(A)</td>
</tr>
<tr>
<td><strong>REGION</strong></td>
<td>OCCURRENCES</td>
</tr>
<tr>
<td><strong>REGION</strong></td>
<td><strong>TOTALS</strong></td>
</tr>
<tr>
<td><strong>REGION</strong></td>
<td><strong>TOTALS</strong></td>
</tr>
<tr>
<td><strong>REGION</strong></td>
<td><strong>TOTALS</strong></td>
</tr>
<tr>
<td><strong>REGION</strong></td>
<td><strong>TOTALS</strong></td>
</tr>
<tr>
<td><strong>REGION</strong></td>
<td><strong>TOTALS</strong></td>
</tr>
<tr>
<td><strong>REGION</strong></td>
<td><strong>TOTALS</strong></td>
</tr>
<tr>
<td><strong>REGION</strong></td>
<td><strong>TOTALS</strong></td>
</tr>
<tr>
<td><strong>REGION</strong></td>
<td><strong>TOTALS</strong></td>
</tr>
<tr>
<td><strong>REGION</strong></td>
<td><strong>TOTALS</strong></td>
</tr>
<tr>
<td><strong>REGION</strong></td>
<td><strong>TOTALS</strong></td>
</tr>
</tbody>
</table>

**Figure 101. Programs-by-Region Report**

## Detecting Database Processing Intent Conflicts

The IMS Monitor records the intervals when a region is in an idle state waiting to update a database owned exclusively by another already scheduled application program.

You can see the total, maximum, and average idle times in the section IDLE FOR INTENT following the DL/I calls. The elapsed time during the unsuccessful scheduling of a program in that region is included in the summary line times for that region.

The region can fail to be scheduled even when ownership of that database is released. The number of times processing is held up by intent failure is separately tallied under the title INTENT FAILURE SUMMARY. The report is illustrated in Figure 102 on page 359. In this report you can see which PSBs are in conflict because of exclusive intent for a segment type and the database name in question.
Examining the Effects of Checkpoints

The checkpoint line of the Region Summary report at the end of the region—by—region summary, shows the following:

- The number of times that a system checkpoint was taken during the monitor interval
- The elapsed times
- The not-wait times

Checkpoint processing can be initiated by the control program at a specified frequency determined by the number of records placed on the system log. Other checkpoints can be caused by operator commands.

The wait time experienced during checkpoints is reported at the end of the first region summary on the Region Wait report. You can detect delays for each combination of DDname and module code. Typical entries here are for the message queue data sets and the restart data set. If a wait for storage is the cause, the entry under the FUNCTION column is STG.= followed by the identification of the pool.

Measuring Region Occupancy

A measure of region activity is the percentage of region occupancy. This is broadly the ratio of the elapsed time a region is performing processing to the trace interval. The region occupancy time does not include those times when no messages are available, when the block loading is delayed, or when the PSB cannot be used. The last section in the Region Summary report lists all active regions for which timed events were collected and shows the calculated percentages of region occupancy.

Monitoring Application Program Elapsed Time

The IMS Monitor can record measurements of elapsed times for each transaction and scheduling of an application program. It does this during the monitored interval while other programs are executing concurrently. Elapsed times are calculated from the start of the first DL/I (or other) call to the end of that program. You can distinguish between time spent in application code and in DL/I processing. Figure 103 on page 360 illustrates the event intervals.
Within the elapsed time for a DL/I call, the wait time to obtain segment data is recorded separately. Similarly, within the elapsed time for an external subsystem call, the processing time in the external subsystem is recorded separately as the wait time. For regions designated as wait-for-input, the time spent waiting for input messages is excluded from values shown in the reports. The application processing (A/P) time includes many kinds of subsidiary service beyond the machine cycles expended by the program object code—such as subroutine loading, I/O to MVS data sets, and any overlay processing. If the program is waiting to be dispatched or requires paging before it can use real storage, these delays are also accounted for in application program processing time. Because a program can execute many transactions for each schedule, the elapsed time from schedule to first call is recorded separately. This time covers the initialization performed by the application program and includes the time for loading the program.

The elapsed times are given in the Program Summary report. Figure 104 on page 361 is an example of the report. Programs are identified by their PSB name on individual lines in the report. Each line gives a summary of the activity for that PSB during the measured interval. The total number of schedules, DL/I calls, transactions completed (dequeued), and waits for DL/I call I/O and external subsystem processing are given. The report line gives calculated average times for:

- Elapsed time per schedule
- Processor time per schedule
- Schedule to first DL/I call per schedule
- Elapsed time per transaction

The report also includes:

- Frequencies for calls per transaction
- I/O waits per DL/I call
- Waits per external subsystem call
- Transactions dequeued per schedule

A TOTALS line summarizes all activity for the PSBs active during the monitored interval. (The PSB DUMMY line reconciles any incomplete scheduling caused by a
You can use the Call-Summary report to examine the detail of the call processing for each program. The report is itemized by type of call and summarized for the monitor interval. An extract from the multipage output is given in Figure 105 on page 362. The calls using an I/O PCB are given first and subtotaled. Then, the total calls of each type, against each database PCB and each external subsystem, are listed. The PSB TOTAL line marks the end of data for each program.
### Monitoring I/O for Application Program DL/I Calls

The IMS Monitor report shows the total number of I/O occurrences and the total time the occurrences took for each application program executed during a monitored interval. The Program-I/O report gives these two totals for all PSBs active during the monitored interval and includes the detailed breakdown of the I/O wait time as it was incurred by each PCB used by the program. Figure 106 on page 364 shows an example of the report.

The report shows any contention experienced during application program processing. Each type of conflict and the number of times it occurred are recorded for each I/O PCB or database PCB. The report shows the total wait time, the highest wait experienced, and the average time. Subtotals are given for each PCB under a PSB, and for all PCBs under each PSB.

The DDN/Func column lists the data set DDName. The Module column uses a code to indicate the source of the contention. The types of conflicts and codes are shown as follows.

- **Message handling**
  - **Code**
    - **DBH**: OSAM I/O for message queues
    - **MFS**: MFS format library directory
    - **PMM**: Message format buffer pool space or control block I/O
Monitoring I/O for Application Program DL/I Calls

QMG  Message queue management

• **Scheduling**

  Code | Conflict
  --- | ---
  BLR  | Load/read from ACBLIB
  MSC  | MPP region initialization
  SMN  | Virtual storage management

• **Database access**

  | Code | Conflict |
  --- | --- |
  DBH  | OSAM I/O |
  DLE  | DL/I functions |
  VBH  | VSAM interface |
  **(Physical segment code)** | Program isolation |

For external subsystem calls, the elapsed time to complete the processing is considered wait time. The **DDN/FUNC** column indicates the external subsystem call function, as shown as follows:

• **External subsystems**

  | Code | Subsystem call function |
  --- | --- |
  AB0  | ABORT |
  CT0  | Create thread |
  D50  | Terminate identify or thread, signoff |
  D80  | INIT |
  I30  | Identify, command, echo, terminate |
  I30  | Identify, terminate subsystem |
  I50  | INIT |
  I60  | Resolve-in-doubt |
  PR0  | Subsystem-not-operational |
  P10  | Commit prepare (Phase 1) |
  P20  | Commit continue (Phase 2) |
  SO0  | Signon |
  SI0  | Identify |
### Figure 106. Program-I/O Report (Part 1 of 2)
The I/O waits for the calls to the I/O PCB, are grouped as the first entries for a PSB. For DL/I calls, the data set for which the I/O took place is indicated under the DDN/FUNC heading, and the module code tells you what type of conflict caused the wait. For external subsystem calls, the function is indicated under the DDN/FUNC heading and the module code indicates the source of the call entry.

**Figure 106. Program-I/O Report (Part 2 of 2)**
Monitoring I/O for Application Program DL/I Calls

Names other than LGMSG and SHMSG can appear in the DDN/FUNC column for I/O PCBs. An example is a checkpoint call issued by an application program (using an I/O PCB) which causes a database buffer to be written.

If the program is designated as wait-for-input and has to wait for the input of the next message, the wait entry is marked "WFI" under the DDN/FUNC heading and no entry appears in the MODULE column. The time spent waiting for the next input message is shown under wait time. "WFI" entries are shown for information only and their values are not used to compute statistics.

Contention for the same physical segment in a database causes a wait on behalf of program isolation. This is shown in the DDN/FUNC column, on the PCB line, by the entry PIdmb, where dmb is the DMB of the physical data set. The MODULE column identifies the segment type using the physical segment code assigned by DBD generation.

When an application is accessing a database using VSAM as the access method, DL/I calls do not generally result in an I/O wait. A MODULE column entry of VBH indicates that interface to VSAM occurred and there was an I/O wait.

A seemingly unrelated entry can occur under the DDN/FUNC column for a database PCB. An example is a retrieval call to a database (DB-A) that causes a buffer to be purged in order to make room for that retrieved data. If the buffer contents included data belonging to another database (DB-B), the I/O entry in the report shows the DDname for DB-B as being in conflict for PCB access to DB-A.

Monitoring MFS Activity

You can obtain a summary of all activity that occurs for management of message format buffer pool use from the Message Format Buffer Pool report. The report is illustrated in Figure 107 on page 367. The data shows the counts at the start and end of the trace interval and their difference.

When message formatting occurs, the appropriate message blocks must reside in the message format buffer pool, a DIF/MID pair for input or a DOF/MOD pair for output. If the blocks are not already in the buffer, I/O to the active IMS.FORMATA/B library must occur. Block retrieval can involve a prior directory lookup, or be direct, using an index kept in the pool.

Many of the counts reveal details of internal event management. When there is no directory entry for a block this implies extra directory lookup I/O.
Monitoring Message Queue Handling

A key resource that directly affects the efficiency of transaction processing is the message queue pool and the management of the I/O to the message queues. You can examine the activity by looking at the Message Queue Pool report. Figure 108 illustrates the report. Counts of activities are given at start and end of the trace interval and as the differences between start and end numbers.

**Figure 108. Message-Queue-Pool Report**
**Detecting Checkpoint Effects**

When a checkpoint command specifies SNAPQ, the current status of all message queues is written to the system log. This prevents any message handling on behalf of queue management. The General Iwait Time Events records the wait time incurred by the SNAPQ. Figure 109 shows the activity on the summary line QMGR SNAPQ CHECK. The number of occurrences is given with the total, average, and maximum wait times.

<table>
<thead>
<tr>
<th>EVENT</th>
<th>OCCURRENCES</th>
<th>TOTAL</th>
<th>MEAN</th>
<th>MAXIMUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>QMGR SNAPQ CHECK</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Figure 109. General Reports for SNAPQ Effects**

**Transaction Queueing Report**

In addition to monitoring the efficiency of message handling, you can monitor the service provided for each application, by looking at the size of the transaction queues at each scheduling of their processing programs.

The Transaction-Queueing report shown in Figure 110 on page 369 records, for each transaction, the minimum, average, and maximum counts at scheduling time. The total number of dequeued transactions (or transactions that have been fully processed) during the monitored interval is given for each transaction code. The average number of transactions processed for each scheduling is given in the DEQUEUED MEAN column.
One of the key resources in an online system is the database buffer pool. The efficiency of DL/I call service depends on the presence of the required database logical record in the buffer, so that segment retrieval does not require additional I/O. This is especially true for “hold” calls with intervening database calls prior to a replace call. You can assess the general efficiency of the pool management using the Database-Buffer-Pool report shown in Figure 111 on page 370. The event counts on this report are not specific to a particular database or program but represent the pressure for use of the database pool.

**Related Reading:** For more information about the Database Buffer Pool reports, see [IMS Version 8: Utilities Reference: Database and Transaction Manager](#).

If any of your databases use VSAM as the access method, the IMS Monitor produces a series of reports headed VSAM BUFFER P00L, one for each subpool. Figure 112 on page 370 shows one of these reports.
### Monitoring Database Buffers

#### DATABASE BUFFER POOL

<table>
<thead>
<tr>
<th>Function</th>
<th>Number of Transactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total buffers in subpool</td>
<td>1000</td>
</tr>
</tbody>
</table>

### Figure 111. Database-Buffer-Pool Report

#### IMS MONITOR*** BUFFER POOL STATISTICS

#### VSAM BUFFER POOL

<table>
<thead>
<tr>
<th>Function</th>
<th>Number of Transactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total buffers in subpool</td>
<td>1000</td>
</tr>
</tbody>
</table>

### Figure 112. VSAM-Buffer-Pool Report

Utilities Reference: System
Monitoring Line Activity

You can obtain a summary of all occurrences of activity for each BTAM line or VTAM node that handles message traffic during the monitored interval. The elapsed times and not-wait times are given in categories of total, mean, and maximum times for each communication line in the Communication-Summary report. Figure 113 illustrates this report.

You must match which physical devices are using the line to the Stage 1 output from system definition. The line numbers are assigned sequentially, according to their physical occurrence in the Stage 1 input deck.

If your online system specifies the prefetch option for MFS blocks in the control region JCL, the last line of the report contains the statistics for all prefetch events.

You can also investigate the amount of data transmitted across BTAM lines or for VTAM nodes with the Line-Functions report. Figure 114 illustrates this report. The report distinguishes between input data and output data. The number of blocks of data and the average and maximum size of the blocks are recorded for data received by IMS and for transmitted data.

This report also includes a measure of how inactive the lines are. An inactive interval is assumed to be the difference between the time that marks the end of the last input block received and the starting time for output transmission. These occurrences of inactivity are termed turnaround intervals, and the report cumulates the number of occurrences as well as the average and maximum times associated with these intervals.

If the line is being used by an MFS-supported terminal, a count of the number of requests for next page for a multipage message is recorded.

If link traffic for coupled multiple systems is recorded, a set of three reports follows the Line Functions report. These are described in the section "Interpreting IMS Monitor MSC Reports" on page 380.

---

**Figure 113. Communication-Summary Report**

<table>
<thead>
<tr>
<th>NODE OR LINE NUMBER</th>
<th>OCCURRENCES</th>
<th>TOTAL</th>
<th>MEAN</th>
<th>MAXIMUM</th>
<th>TOTAL</th>
<th>MEAN</th>
<th>MAXIMUM</th>
<th>DISTRIBUTION NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMTO1A</td>
<td>3</td>
<td>2396</td>
<td>798</td>
<td>1547</td>
<td>2396</td>
<td>798</td>
<td>1547</td>
<td>1467A,B</td>
</tr>
<tr>
<td>19</td>
<td>182</td>
<td>92155</td>
<td>506</td>
<td>1106</td>
<td>92155</td>
<td>506</td>
<td>1106</td>
<td>1493A,B</td>
</tr>
<tr>
<td>2</td>
<td>59</td>
<td>2280</td>
<td>38</td>
<td>41</td>
<td>2280</td>
<td>38</td>
<td>41</td>
<td>1515A,B</td>
</tr>
<tr>
<td>TOTAL</td>
<td>244</td>
<td>96831</td>
<td>396</td>
<td></td>
<td>96831</td>
<td>396</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 114. Line-Functions Report**

<table>
<thead>
<tr>
<th>NODE OR LINE NUMBER</th>
<th>RECEIVE BLOCKS</th>
<th>RECEIVE BLKSIZE</th>
<th>MAX. TRANS. BLOCKS</th>
<th>BLKSIZE</th>
<th>MAX. TRANS. DIST. NUMBER</th>
<th>TURN AROUND INTERVAL</th>
<th>MEAN</th>
<th>MAX. DIST. NUMB.</th>
<th>PAGING REQUESTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMTO1A</td>
<td>1</td>
<td>29</td>
<td>2</td>
<td>170</td>
<td>171</td>
<td>3</td>
<td>798</td>
<td>1547</td>
<td>1466</td>
</tr>
<tr>
<td>19</td>
<td>182</td>
<td>506</td>
<td>1106</td>
<td>1492</td>
<td>1514</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>59</td>
<td>38</td>
<td>41</td>
<td>1514</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALL LINES</td>
<td>1</td>
<td>29</td>
<td>2</td>
<td>170</td>
<td></td>
<td>244</td>
<td>396</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
Monitoring Message Handling Efficiency

The IMS Monitor produces both summary and detailed information on asynchronous processing that takes place in the IMS control region. Asynchronous processing occurs when data transmitted from BTAM terminals or from VTAM arrives. Application program responses also result in asynchronous processing.

The space in four major buffer pools and access to format, SPA, and message queue data sets are managed for the total communications traffic. Wait times are recorded when contention for pool space or I/O interrupts the processing of any of the communication tasks triggered by line activity. This information is contained in the Communication Wait report, shown in Figure 115.

This report is similar to the Communication-Summary report because the line number identifies the series of communication processing tasks.

Pool Space Failure

The Pool Space Failure Summary report gives the number of times in each region a given amount of storage was unavailable. It shows the number of bytes, the identification of the pool, and the number of occurrences when storage was unavailable. You can use this summary to determine if you need to increase the buffer pool allocation, either by a system definition change or by overriding the number of buffers in the EXEC statements in the JCL.

The format of the report is shown in Figure 116.

Programs Experiencing Deadlock

Each time a pair of programs reaches a deadlock over ownership of a segment in a given database data set, the Deadlock-Event-Summary report records the occurrence. Each line in the report shows the two PSBs involved and indicates which is given processing right-of-way (REQ-ING PSB) and which has to reprocess...
after dynamic backout occurs (LOSING PSB). The report is illustrated in Figure 117.

**DEADLOCK EVENT SUMMARY**

<table>
<thead>
<tr>
<th>REQ-ING PSB</th>
<th>LOSING PSB</th>
<th>DBNAME</th>
<th>OCCURRENCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSBNAME1</td>
<td>TPPSRE3</td>
<td>DBASEBAL</td>
<td>1</td>
</tr>
<tr>
<td>TOTAL</td>
<td>TOTAL</td>
<td>TOTAL</td>
<td>TOTAL</td>
</tr>
</tbody>
</table>

*Figure 117. Deadlock-Event-Summary Report*

**IMS Latch Conflict**

The basic serialization of the task processing in IMS is controlled by ownership of an IMS latch. When different programs are executing, they compete for the ownership. If they wait for the resource, the one possessing the latch has to post the other ITASK waiting for it. Use the Latch Conflict Statistics report to judge the level of contention for a resource.

The different types of latches and the counters that exhibit the level of contention are given in the Latch Conflict Statistics report. [Figure 118 on page 375](#) is an example of this report. The entries are organized according to the latch names.

The latch names, which are abbreviations for the different types of resources being serialized, are as follows:

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Latch Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>'DISP'</td>
<td>SYS/DISPATCHER</td>
</tr>
<tr>
<td>'DCSL'</td>
<td>DC/CHECKPOINT DC SYSTEM</td>
</tr>
<tr>
<td>'LUML'</td>
<td>DC/LU 6.2 LUM</td>
</tr>
<tr>
<td>'CONV'</td>
<td>DC/CONVERSATION CHECKPT</td>
</tr>
<tr>
<td>'TERM'</td>
<td>DC/TERMINAL</td>
</tr>
<tr>
<td>'LUBT'</td>
<td>DC/LU62 LUB-TIB CHAIN</td>
</tr>
<tr>
<td>'SCHD'</td>
<td>TM/SCHEDULING</td>
</tr>
<tr>
<td>'TCTB'</td>
<td>TM/TCT BLOCK</td>
</tr>
<tr>
<td>'APSB'</td>
<td>TM/ALLOCATE PSB (BLK MVR)</td>
</tr>
<tr>
<td>'PDRB'</td>
<td>TM/PDIR BLOCK (BLK MVR)</td>
</tr>
<tr>
<td>'PSBP'</td>
<td>TM/PSB POOL (BLK MVR)</td>
</tr>
<tr>
<td>'DMBP'</td>
<td>TM/DMB POOL (BLK MVR)</td>
</tr>
<tr>
<td>'PSBB'</td>
<td>TM/PSB BLOCK (BLK MVR)</td>
</tr>
<tr>
<td>'DMBB'</td>
<td>TM/DMB BLOCK (BLK MVR)</td>
</tr>
<tr>
<td>'PDRP'</td>
<td>TM/PDIR POOL (BLK MVR)</td>
</tr>
<tr>
<td>'DBAU'</td>
<td>TM/DBRC AUTH (BLK MVR)</td>
</tr>
<tr>
<td>'DDRBP'</td>
<td>TM/DDIR BLOCK (BLK MVR)</td>
</tr>
<tr>
<td>'DDRP'</td>
<td>TM/DDIR POOL (BLK MVR)</td>
</tr>
<tr>
<td>'BBBP'</td>
<td>DB/OSAM BUFFER POOL</td>
</tr>
<tr>
<td>'DBLR'</td>
<td>DB/DFSDBLR0 MODULE</td>
</tr>
</tbody>
</table>
When a system checkpoint is taken during the time the monitor is active, latch conflict statistics are reset to zero, thus corrupting the values presented in this report. If this situation exists, the following message will be inserted at the top of the report:

**** A CHECKPOINT OCCURRED DURING MONITOR RUN ****
**** LATCH CONFLICT STATISTICS ARE INVALID ****
**** SEE UTILITIES REFERENCE MANUAL ****

However, if the master terminal operator issues the /CHECKPOINT command with the STATISTICS keyword parameter, latch conflict statistics are reset to zero, but the IMS monitor is not notified. Therefore, DFSUTR20 cannot detect that the statistics have been corrupted and does not issue this message.

**Recommendation:** Do not issue statistics checkpoints while the monitor is running.
Using Frequency Distributions from IMS Monitor Output

The reports derived from the IMS Monitor data records contain many summary lines where the mean time is given. If you are interested in the distribution of those timed events, rather than just average and maximum times, you can request the Report Print utility to individually record the events in a frequency distribution across a range of intervals. Some distributions are not time dependent, such as those for transaction queue loads or transmitted block sizes.

How to Get a Frequency Distribution Output

To request the IMS Monitor Report Print utility to gather distribution data, you must include a DIS input control statement. This causes all report items with an entry under a column headed MEAN to have a corresponding frequency distribution as part of the Distribution Appendix. Each report line includes an identifying reference number under the column headed Distribution Number so that you can locate the distribution data in the appendix, flagged by that same number.

The following tables show the major IMS Monitor reports and the type of frequency distributions generated for each report. Each type results in several distributions, depending on how many entries are in each section of the report. For each type of frequency distribution, the data is cumulated in suitable intervals or ranges. The set of ranges used for each type is given an identifier, shown in the ID column. Table 23 shows the report distributions sorted by Region Summary.

Table 23. Distribution Reports by Region Summary

<table>
<thead>
<tr>
<th>Report Name</th>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule end to 1st DL/I call</td>
<td>D1</td>
<td>Elapsed time</td>
</tr>
<tr>
<td>Elapsed execution time</td>
<td>D2</td>
<td>Not wait time</td>
</tr>
<tr>
<td>DL/I calls</td>
<td>D3</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>D4</td>
<td>N/A</td>
</tr>
<tr>
<td>External Subsystem calls</td>
<td>D5</td>
<td>Elapsed time</td>
</tr>
<tr>
<td>Waits per DL/I call</td>
<td>D6</td>
<td>Not wait time</td>
</tr>
<tr>
<td>Idle for intent</td>
<td>D43</td>
<td>Elapsed time</td>
</tr>
</tbody>
</table>
### Frequency Distribution

**Table 23. Distribution Reports by Region Summary (continued)**

<table>
<thead>
<tr>
<th>Report Name</th>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Checkpoint</td>
<td>D7</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>D8</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>D20</td>
<td>Elapsed time</td>
</tr>
<tr>
<td></td>
<td>D21</td>
<td>Not wait time</td>
</tr>
</tbody>
</table>

**Table 24** shows the report distributions sorted by Program Region.

**Table 24. Report Distributions by Program Region**

<table>
<thead>
<tr>
<th>Report Name</th>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elapsed execution time</td>
<td>D30</td>
<td>N/A</td>
</tr>
<tr>
<td>Schedule end to 1st DL/I</td>
<td>D31</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Table 25** shows the report distributions sorted by Program Summary.

**Table 25. Report Distributions by Program Summary**

<table>
<thead>
<tr>
<th>Report Name</th>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor time per schedule</td>
<td>D15</td>
<td>N/A</td>
</tr>
<tr>
<td>Transactions dequeued per schedule</td>
<td>D14</td>
<td>N/A</td>
</tr>
<tr>
<td>Elapsed time per schedule</td>
<td>D9</td>
<td>N/A</td>
</tr>
<tr>
<td>Schedule end to 1st DL/I call</td>
<td>D10</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Table 26** shows the report distributions sorted by Communication Summary.

**Table 26. Report Distributions by Communication Summary**

<table>
<thead>
<tr>
<th>Report Name</th>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line elapsed time</td>
<td>D18</td>
<td>N/A</td>
</tr>
<tr>
<td>Line not wait time</td>
<td>D19</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Table 27** shows the report distributions sorted by Line Functions.

**Table 27. Report Distributions by Line Functions**

<table>
<thead>
<tr>
<th>Report Name</th>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Received block length</td>
<td>D36</td>
<td>N/A</td>
</tr>
<tr>
<td>Transmitted block length</td>
<td>D37</td>
<td>N/A</td>
</tr>
<tr>
<td>Inactive intervals</td>
<td>D38</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Table 28** shows the report distributions sorted by MSC Queuing Summary.

**Table 28. Report Distributions by MSC Queuing Summary**

<table>
<thead>
<tr>
<th>Report Name</th>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time in queue</td>
<td>D39</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Table 29 shows the report distributions sorted by Transaction Queuing.

Table 29. Report Distributions by Transaction Queuing

<table>
<thead>
<tr>
<th>Report Name</th>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transactions on queue at schedule</td>
<td>D17</td>
<td>N/A</td>
</tr>
<tr>
<td>Transactions dequeued per schedule</td>
<td>D16</td>
<td>N/A</td>
</tr>
<tr>
<td>Prefetch format blocks</td>
<td>D28</td>
<td>Elapsed time</td>
</tr>
<tr>
<td></td>
<td>D29</td>
<td>Not wait time</td>
</tr>
</tbody>
</table>

Table 30 shows the report distributions sorted by Call Summary.

Table 30. Report Distributions by Call Summary

<table>
<thead>
<tr>
<th>Report Name</th>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSB waits per DL/I call</td>
<td>D13</td>
<td>N/A</td>
</tr>
<tr>
<td>PSB waits per external subsystem call</td>
<td>D44</td>
<td>N/A</td>
</tr>
<tr>
<td>PSB elapsed time per call</td>
<td>D11</td>
<td>N/A</td>
</tr>
<tr>
<td>PSB not wait time per call</td>
<td>D12</td>
<td>N/A</td>
</tr>
<tr>
<td>PSB external subsystem calls</td>
<td>D45</td>
<td>Elapsed time</td>
</tr>
</tbody>
</table>

Table 31 shows some distributions derived from buffer pool statistics for wait times.

Table 31. Wait Time Distributions

<table>
<thead>
<tr>
<th>Function</th>
<th>ID</th>
<th>Module Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage</td>
<td>D22</td>
<td>SMN</td>
</tr>
<tr>
<td>OSAM I/O</td>
<td>D23</td>
<td>DBH</td>
</tr>
<tr>
<td>VSAM I/O</td>
<td>D24</td>
<td>VBH</td>
</tr>
<tr>
<td>Scheduler internal</td>
<td>D25</td>
<td>MSC</td>
</tr>
<tr>
<td>Queue manager I/O</td>
<td>D26</td>
<td>QMG</td>
</tr>
<tr>
<td>Block loader I/O</td>
<td>D27</td>
<td>BLR</td>
</tr>
<tr>
<td>MFS block I/O</td>
<td>D32</td>
<td>MFS</td>
</tr>
<tr>
<td>MFS directory I/O</td>
<td>D33</td>
<td>MFS</td>
</tr>
<tr>
<td>HSAM I/O</td>
<td>D34</td>
<td>DIE</td>
</tr>
<tr>
<td>Format Buffer Pool Space</td>
<td>D35</td>
<td>PMM</td>
</tr>
<tr>
<td>PI enqueue</td>
<td>D40</td>
<td>None</td>
</tr>
<tr>
<td>QMGR SNAPQ Check</td>
<td>D42</td>
<td>None</td>
</tr>
</tbody>
</table>

How Frequency Distribution Ranges Are Defined

A set of ten intervals is defined for each summary line and the occurrences falling in each interval are cumulated. The interval ranges are preset with default end points. For example, the end points, for DL/I call elapsed time are: 0, 1000, 2000, 4000, 8000, 16000, 32000, 64000, 128000, 256000, INF (all times are in milliseconds). The default end points are chosen so that they are suitable to the event. The lower limit of the first interval always defaults to zero, and the upper limit of the tenth interval is infinity (INF).
Frequency Distribution

Although several types of distribution can use the same set of end points, each type is assigned a distribution identifier. You can use this to redefine the end points. To override the default end points, include an input control statement to the Report Print utility. The statement specifies the type of distribution identifier and gives the desired end point values. For example, the DL/I call elapsed time end points could be respecified by:

D5  0,500,1000,1500,2000,4000,,100000,500000

The values of the unspecified end points remain at their default values of 32000 and 64000 as does the last (INF).

Figure 119 which is a sample page taken from a Distribution Appendix, shows how individual distributions are numbered, and how ranges vary with the type of distribution. The lines are arranged in pairs, with the second one recording the cumulated counts.

Utilities Reference: System

**DISTRIBUTION***

**D3, D28,**

**locating it in the following list:**

You can determine the default end points for the distribution by 375 through Table 30 on page 377) and the Wait Time Distributions table (Table 31 on page 377) you can determine the default end points for the distribution by

locating it in the following list:

- **D1, D2, D5, D6, D9, D10, D11, D12, D15 D18, D19, D20, D21, D22, D25, D27, D28, D29, D30, D31, D43, and D45**
  - 0, 1000, 2000, 4000, 8000, 16000, 32000, 64000, 128000, 256000, INF
  - 0, 50000, 100000, 150000, 200000, 250000, 300000, 350000, 400000, 450000, INF
  - 0, 200000, 400000, 600000, 800000, 1000000, 1200000, 1400000, 1600000, 1800000, INF
D7, D13, and D44
0, 0, 1, 2, 3, 4, 5, 6, 7, 8, INF
D8
0, 100000, 200000, 300000, 400000, 500000, 600000, 700000, 800000, 900000, INF
D14, D16, D17
0, 1, 2, 3, 4, 5, 10, 15, 30, 90, INF
D23, D24, D26, D32, D40, D42
0, 2000, 8000, 24000, 50000, 100000, 150000, 200000, 250000, 300000, INF
D33, D34, D35
0, 2000, 4000, 8000, 16000, 32000, 64000, 96000, 128000, 160000, INF
D36, D37
0, 10, 20, 40, 80, 100, 200, 400, 800, 1000, INF
D38
0, 1000, 10000, 100000, 200000, 500000, 1000000, 1500000, 2000000, INF
D39
0, 1000, 5000, 10000, 50000, 100000, 500000, 1000000, 5000000, 10000000, INF

Interpreting Distribution Appendix

You can use the detailed output in the Distribution Appendix when you suspect an unusual combination of events was reported in a report summary line. Usually, the average and maximum times or counts are sufficient to highlight a resource usage problem. However, if you suspect the mean value to be masking an unusual distribution you can draw on the detail contained in the IMS Monitor output records.

For example, suppose you are investigating a change in the scheduling algorithm for a particular transaction and need to know how many transactions were able to be processed for each scheduling of an application program. Figure 120 shows a possible histogram for the processed transactions:

![Figure 120. Number ofTransactions Processed For Each Scheduling Of An Application Program](image)

The Average=2.5 transactions per schedule. The distribution in Figure 120 suggests that many schedules were able to process only one or two transactions, and few schedules significantly exhausted the queue. The distribution data for the histogram is as follows:
Frequency Distribution

<table>
<thead>
<tr>
<th>Number of schedules</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6-10</th>
<th>&gt;10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transactions dequeued</td>
<td>8</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

The Distribution Appendix presents the histogram data in the form of two lines:
- The first line shows the intervals, prefixed by a cross reference to an individual line on the earlier output.
- The second line gives the number of events occurring in those intervals.

This data appears as follows:

<table>
<thead>
<tr>
<th>#</th>
<th>950B...0...1...2...3...4...5...10...15...30...90...INF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8</td>
</tr>
</tbody>
</table>

The cross reference 950B points to a unique report line. For example, the Transaction-Queuing report on the appropriate line for the transaction of interest show, 950A,B under the column headed DISTRIBUTION NUMBER. Use the reference number 950B to locate the data in the Distribution Appendix. The 950A reference points to the data for the number of transactions in the queue at schedule time.

Interpreting IMS Monitor MSC Reports

The IMS Monitor Report Print program includes three reports that highlight message events caused by system coupling.
- MSC-Traffic report
  - This report shows the enqueue and dequeue counts of messages that use the various link paths for the monitor interval.
- MSC-Summary report
  - This report shows summaries of:
    - The traffic queues for each input transaction name
    - The traffic queues for each destination name
    - The traffic queues for each link number
    - The traffic queues for each destination system
- MSC-Queuing-Summary report
  - This report is generated only when intersystem messages are queued on the local system before being sent to the destination system. The local system must be an intermediate system. This report shows:
    - Maximum time messages spend in queues
    - Average time messages spend in queues
    - Maximum queue lengths
    - Maximum queue counts
    - Total number of messages queued for all links the local system participates in

All three of the reports can have entries in the Distribution Appendix. You can examine the frequency distributions of the traffic if you suspect unusual transmission patterns.

Determining Cross-System Queuing

The MSC-Traffic report reveals the individual queue loads for all traffic between partner systems of which the monitored system is the local system. The report lists all the unique system identification numbers (SIDs) that are defined for communications for that local system. It then summarizes the total messages queued and dequeued for each combination of the following variables:
Assessing

- Input name (terminal or program that was a message source)
- Destination name (terminal or program)
- Input system (SID)
- Destination system (SID)
- Link number
- Link type (BSYN, MTM, CTC, or VTAM)

Figure 121 illustrates this report. If a message originates in the local system, its presence is accounted for in the dequeue counts only. Messages with local destinations appear only in the enqueue count.

<table>
<thead>
<tr>
<th>LOCAL SID</th>
<th>INPUT NAME</th>
<th>INPUT VALUE</th>
<th>DESTIN. NAME</th>
<th>DESTIN. VALUE</th>
<th>LINK NO.</th>
<th>LINK TYPE</th>
<th>ENQUEUE COUNT</th>
<th>DEQUEUE COUNT</th>
</tr>
</thead>
</table>
| **Figure 121. MSC-Traffic Report**

Assessing the Effect of Link Loading

The MSC-Summary report shows you the enqueue and dequeue activity for messages that are handled by the local system but are part of the multiple system coupling traffic. The report format is illustrated in Figure 122 on page 382.
Interpreting IMS Monitor MSC Reports

The first set of queuing counts shows how many transactions of each type were entered in the monitor interval, and how many were subsequently dequeued.

The second set of counts summarizes the total traffic for each destination name. You can distinguish the primary transactions and responses by the resource names and examine the relative servicing of the link transmissions using the difference between the enqueue and dequeue counts.

The third set of counts lists the active links by link number, and you can determine if there is buildup on the link by the difference in the enqueue and dequeue counts.

The fourth set of counts records the traffic that is going to other systems by all link paths. You can judge by the difference in enqueue and dequeue counts whether the overall pattern of link priorities to one or more systems is causing buildup of cross-system traffic.

<table>
<thead>
<tr>
<th>INPUT NAME</th>
<th>ENQUEUE CNT</th>
<th>DEQUEUE CNT</th>
<th>DESTIN. NAME</th>
<th>ENQUEUE CNT</th>
<th>DEQUEUE CNT</th>
<th>LINK NO.</th>
<th>ENQUEUE CNT</th>
<th>DEQUEUE CNT</th>
<th>DEST. SID</th>
<th>ENQUEUE CNT</th>
<th>DEQUEUE CNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSTM3577</td>
<td>1</td>
<td>1</td>
<td>DSTM4358</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM4048</td>
<td>1</td>
<td>1</td>
<td>DSTM4358</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM5216</td>
<td>1</td>
<td>1</td>
<td>DSTM5457</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM5496</td>
<td>1</td>
<td>1</td>
<td>DSTM5332</td>
<td>0</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM5277</td>
<td>1</td>
<td>1</td>
<td>DSTM5604</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM5711</td>
<td>1</td>
<td>1</td>
<td>DSTM3347</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM5274</td>
<td>1</td>
<td>1</td>
<td>DSTM5338</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM5807</td>
<td>1</td>
<td>1</td>
<td>DSTM3268</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM3685</td>
<td>1</td>
<td>1</td>
<td>DSTM3676</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM6161</td>
<td>1</td>
<td>1</td>
<td>DSTM5428</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM5838</td>
<td>1</td>
<td>1</td>
<td>DSTM5395</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM4638</td>
<td>1</td>
<td>1</td>
<td>DSTM4168</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM3903</td>
<td>1</td>
<td>1</td>
<td>DSTM3056</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM5418</td>
<td>1</td>
<td>1</td>
<td>DSTM3511</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM4673</td>
<td>1</td>
<td>1</td>
<td>DSTM3363</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM5141</td>
<td>1</td>
<td>1</td>
<td>DSTM3674</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM4391</td>
<td>1</td>
<td>1</td>
<td>DSTM4467</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM3324</td>
<td>1</td>
<td>1</td>
<td>DSTM4501</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM4781</td>
<td>1</td>
<td>1</td>
<td>DSTM5037</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM3525</td>
<td>1</td>
<td>1</td>
<td>DSTM4298</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM4542</td>
<td>1</td>
<td>1</td>
<td>DSTM5778</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM5796</td>
<td>1</td>
<td>1</td>
<td>DSTM4003</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM4782</td>
<td>1</td>
<td>1</td>
<td>DSTM3988</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM4633</td>
<td>1</td>
<td>1</td>
<td>DSTM4217</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM3655</td>
<td>1</td>
<td>1</td>
<td>DSTM6135</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM3892</td>
<td>1</td>
<td>1</td>
<td>DSTM5147</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM3338</td>
<td>1</td>
<td>1</td>
<td>DSTM5381</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM4681</td>
<td>1</td>
<td>1</td>
<td>DSTM5593</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM4862</td>
<td>1</td>
<td>1</td>
<td>DSTM3304</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM4902</td>
<td>1</td>
<td>1</td>
<td>DSTM5081</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTM4558</td>
<td>1</td>
<td>1</td>
<td>DSTM4671</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DSTM3655</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DSTM3892</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DSTM3338</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DSTM4681</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DSTM4482</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DSTM4902</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DSTM4558</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DSTM3926</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 122. MSC-Summaries Report**

Assessing Link Queuing Times

The MSC-Queuing-Summary report provides information about intersystem message traffic only. You can use the sample of traffic recorded in the IMS Monitor interval to examine the maximum and average time messages spend in queues waiting to be sent on active links. You can detect whether the link priorities are causing undue delay of primary messages through the intermediate system, or
whether there is a build up of responses. The report shows the logical link paths for this system which is an intermediate system. Each incoming link number shows the number of messages that are queued before transmission on their specified outward bound link number. The maximum queue count is given as well as the maximum and average time on the intermediate system queues.

The report is illustrated in Figure 123.

---

**Extracting Multiple System Transaction Statistics**

You can use the Log Transaction Analysis utility to obtain counts of the message traffic both in local systems and between systems. The transmissions over the different types of physical links can also be examined. The activity is summarized for each step of the logical link paths. You must provide IMS system log input that reflects all partner system activity, that is, sets of system logs for each MSC system. To coordinate the sets of individual system logs use the Log Merge utility. Up to nine separate system logs can be merged; each log must be the output of a uniquely identified IMS system with MSC installed.

**Controlling the Log Merge**

To control the log output, you must:

- Choose the required systems that take part in the logical link paths you are examining.
- Coordinate the series of input logs for each system so that they cover a similar time span.
- Specify a start and stop time for the Log Merge utility control statements if you are sampling the cross-system processing for a particular 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 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.)

**Interpreting the Transaction Analysis Report**

You can use the Log Analysis report produced by the Log Transaction Analysis utility to obtain the following statistics for individual transactions processed in any system:

- The total response time
- The time on input and output queues
- The processing time
Chapter 21, “Interpreting Statistical-Analysis and Log-Transaction Reports,” on page 455 defines the format of the detailed report records produced by this utility, provides a list of processing type codes, and shows an illustration of the report. The absence of times for a message GU call or MPP termination in the report lines indicate an input source or intermediate system report line.

The processing type field is an important one for the interpretation of the detailed report lines. The S code indicates that this line shows a send or receive event for the transaction. You can trace the progress of a cross-system conversation using the codes C, D, P, X, and Y.

The report headings include a column headed ID after the column for the GU to the message queue time. The number shown in a report line under the ID heading matches the sequence in which log input was fed to the Log Merge utility. The field corresponds to starting position 102, the 3-digit field named SYSTEM ID, in the detailed report records.

You can use the sort step to order the report records by system ID within transaction code, or other convenient sequences, rather than by the default of the overall input sequence.
This chapter describes:
- The events that the IMS Monitor collects
- The content of the reports produced by the IMS Monitor Report Print Program

Monitoring has different meanings for DBCTL and DB/DC. For DB/DC, the end user enters the transaction on a terminal. The transaction is processed by IMS and then returns a result to the user. Transaction characteristics that are monitored include total response time and the occurrences of resource contentions (for example, PSB schedule wait time, and database I/Os).

DBCTL, on the other hand, has neither transactions nor terminal end users. It does, however, work on behalf of transactions entered by CCTL terminal users. DBCTL monitoring provides data about the processing that occurs when a CCTL transaction accesses DBCTL databases. The CCTL gains this access using DRA requests.

A typical sequence of these DRA requests would be:
1. A SCHED request to get a PSB scheduled in DBCTL
2. A DL/I request to make database calls
3. A sync-point request, COMMTERM, to commit the updates and release the PSB

The DBCTL process that encompasses these requests is called a unit of recovery (UOR).

DBCTL provides monitoring data about UORs, such as: total time UOR existed, wait time for PSB schedule, and I/Os during database calls. This information is very similar to IMS transaction monitor data. In a DBCTL-CCTL system, however, the UOR data represents only part of the total processing of a CCTL transaction. Therefore, CCTL monitor data is necessary to get a total view of CCTL transaction performance.

DBCTL does not change the format or usage of the IMS monitor reports. There are reports and fields within reports that are not applicable to DBCTL. Generally, these are in the transaction manager and communication areas. There are some fields that are interpreted differently in a DBCTL environment.

For reports that do not apply to DBCTL, either a heading without data is shown or no report is generated. These reports are:
- Message Queue Pool report
- Message Format Buffer Pool report
- Communication Summary report
- Communication IWAIT report
- Line Functions report
- MSC Traffic report
- MSC Summaries report
- MSC Queuing Summary report

The term region in IMS Monitor reports refers to a PST assigned to a specific dependent region that processes specific IMS transactions. In DBCTL monitor reports the term region still applies to a PST. A PST can service one CCTL thread (transaction) at a time. However, CCTL threads change, resulting in one PST
servicing many different CCTL transactions. Since multiple CCTLs can connect to DBCTL, the PST can actually service transactions from different CCTLs.

All of the threads built for a CCTL carry the job name of the CCTL. This appears as the same job name for many regions in the General Reports.

Within a trace interval, a thread can be assigned to multiple CCTLs, but it can only be assigned to one CCTL at any instant of time. So, depending on the number of CCTLs attached to DBCTL, the Region Summary reports can show:

- One region with only one job name.
- One region with different job names.
- Multiple regions with different job names. Some regions can have the same job name and some can have different job names.
- Multiple regions with only one job name.

Any monitor report for a region is a summary of all the CCTLs a thread served during the trace interval (for example, the elapsed time for all CCTLs that a thread has been assigned to during the trace interval).

The reports generated by the IMS Monitor are the same for BMPs and non-message BMPs.

UOR elapsed times are spent in DBCTL, not in the DRA. The time spent in the DRA is considered part of the CCTL, therefore the DRA time is not reported by any DBCTL statistics.

In this Chapter:

- “IMS Monitor Trace Event Intervals”
- “Overview of IMS Monitor Reports” on page 387
- “Documenting the Monitoring Run” on page 388
- “Monitoring Activity in Dependent Regions” on page 390
- “Monitoring Application Program Elapsed Time” on page 395
- “Monitoring Database Buffers” on page 399
- “IMS Internal Resource Usage” on page 401
- “Using Frequency Distributions from IMS Monitor Output” on page 402

**IMS Monitor Trace Event Intervals**

The IMS Monitor trace interval is defined by the master terminal operator’s use of the /TRACE command between the start and stop command entries. The online IMS events are recorded in IMS Monitor records placed in the IMSMON data set. The event timings are related to dependent region activity. Figure 124 on page 387 shows the boundaries of the timed event intervals.

The Monitor trace interval includes the following intervals:

- Scheduling and Termination
  - Block loader busy
  - Intent failures (exclusive intent and data sharing) and Schedule failures (PSB busy and space failure)
  - Sched/Term elapsed
  - NOT-WAIT
- ACBLIB waits
- DB Flush waits
- DB CLOSE waits

- Region occupancy (which overlaps with all of Sched/Term elapsed)
  - Schedule to first call
  - Elapsed execution

The NOT-WAIT time for a region is the elapsed time not accounted for by wait time. Any delay coming from either paging or the processor being dispatched for a higher priority task results in an increase in the NOT-WAIT times.

![IMS Monitor Trace Event Intervals](image)

**Figure 124. IMS Monitor Trace Event Intervals**

### Overview of IMS Monitor Reports

A list of reports available from data collected by the IMS Monitor, together with the principal performance data they contain, is shown in Table 32.

**Related Reading:** For a description of those reports marked “DB”, see [IMS Version 8: Utilities Reference: Database and Transaction Manager](#).

### Sequence of Report Output

The order of the reports listed in Table 32 matches the sequence of the output from the IMS Monitor Report Print program. The duration of a monitoring snapshot might not include certain events necessary for an individual report, in which case only report headings or partial data are produced.

**Table 32. IMS Monitor Reports Output Sequence and Information**

<table>
<thead>
<tr>
<th>Report Name</th>
<th>Principal Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Configuration</td>
<td>Monitor run documentation</td>
</tr>
<tr>
<td>Database Buffer Pool (DB)</td>
<td>Count of DB calls and I/O per transaction</td>
</tr>
<tr>
<td>VSAM Buffer Pool (DB)</td>
<td>Count of inserts and I/Os</td>
</tr>
<tr>
<td>Latch Conflict Statistics</td>
<td>IMS internal processing</td>
</tr>
<tr>
<td>Region and Jobname</td>
<td>Monitor run documentation</td>
</tr>
<tr>
<td>Region Summary</td>
<td>Elapsed times and count of DL/I calls</td>
</tr>
<tr>
<td>Region Wait</td>
<td>Wait times</td>
</tr>
</tbody>
</table>
### Overview of IMS Monitor Reports

<table>
<thead>
<tr>
<th>Report Name</th>
<th>Principal Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programs by Region</td>
<td>Elapsed times for region usage</td>
</tr>
<tr>
<td>Program Summary</td>
<td>Overall program statistics</td>
</tr>
<tr>
<td>Program I/O (DB)</td>
<td>Wait times/PCB</td>
</tr>
<tr>
<td>Reports</td>
<td>Count of space failures and deadlocks</td>
</tr>
<tr>
<td>Run Profile</td>
<td>Monitor run documentation</td>
</tr>
<tr>
<td>Call Summary (DB)</td>
<td>Call counts and timings/segment type</td>
</tr>
<tr>
<td>Distribution Appendix</td>
<td>Event frequency distributions</td>
</tr>
</tbody>
</table>

#### Units of Measure in IMS Monitor Reports

The majority of the data items in IMS Monitor reports are elapsed times. These are normally expressed in microseconds. An entry of 1876534 is 1.876534 seconds or 1876 milliseconds. Any times that do not follow this convention show the unit of measure on the report.

You can also find counts of events under the heading OCCURRENCES, and some figures that represent the number of bytes.

#### Documenting the Monitoring Run

For each trace interval, several general reports or overall summaries are generated for the processing that took place. You can use these reports as part of your IMS Monitor run documentation.

It is important to record, as accurately as possible, the conditions under which the trace was taken. Your documentation can include system status information (obtained by the /DISPLAY command) several times before and after the trace, an expected profile of the CCTL transaction activity, and any desired processing events. The trace interval should represent typical processing loads and not be a biased or inadequate historical record.

#### Adding to the System Configuration Report Data

The first general report (titled SYSTEM CONFIGURATION) is found under the page heading BUFFER POOL STATISTICS. It shows the modification level of the IMS and MVS systems. You can choose to add a list of IMS APARs applied and include the service levels of the application programs, especially if the latter are not permanent programs or are part of a staged implementation. The system configuration output is illustrated in Figure 125 on page 389.

#### Recording the Monitor Trace Interval

The heading of most IMS Monitor reports carries the trace start and stop times. It is shown in the format YEAR DAY (Julian) HH:MM:SS. The overall length of the trace interval is given in milliseconds under the title MONITOR OVERHEAD DATA. The following line shows how many trace records were placed on the IMSMON data set. An example of the monitor trace interval recording is shown in Figure 125 on page 389.
Completing the Monitor Run Profile

A compact set of processing ratios will be found at the end of the Run Profile report. The statistics summarize, for the monitor interval, the UOR throughput and the degree of DL/I and I/O activity. An example of the report is shown in Figure 126.

The lower half of the Run Profile report shows several ratios:
- Program elapsed time to DL/I elapsed time for each region
- DL/I elapsed time to wait time during DL/I processing
- Program elapsed time to other subsystem call elapsed time
- DL/I elapsed time to other subsystem call elapsed time

Each region is identified by a sequence number, starting at region 1.

There are some generalized processing ratios that are given at the end of several buffer pool statistics reports. You can include them in the documented profile of the trace interval. These are not specific to one UOR or system resource but can be used as indicators of variation across a series of monitor runs. The ratios are:
Documenting the Monitor Run

- The total number of OSAM reads + OSAM writes + all waits divided by the total number of transactions.
  From the Message Queue Pool report, this ratio indicates on a per transaction basis the physical I/O activity required to handle the scheduling function.
- The total number of OSAM reads + OSAM writes + BISAM reads divided by the total number of transactions.
  From the Database Buffer Pool report, this ratio indicates on a per transaction basis the physical I/O activity required to handle the database buffering function.

Verifying IMS Monitor Report Occurrences

When you examine the output from the IMS Monitor Report Print Program, the presence of a report heading does not necessarily mean that appropriate data will be listed. System definition options and utility control statements also affect the content of the output as follows:

- The output does not include a Call Summary report unless a control statement specifies DLI.
- The output does not include a set of Distribution reports unless a control statement specifies DIS or DISTRIBUTION. The column headed DISTRIBUTION NUMBER that occurs on many of the reports contains cross-references to items included in the Distribution reports.
- The output consists of just a Call Summary report if a control statement specified ONLY DLI.

Because many of the summary reports require system status to calculate the difference between start and end values, and this status is obtained during the /TRACE SET OFF processing, the IMS Monitor execution must end before termination of the IMS control region. If the trace was not stopped properly, the following message is issued:

NO DATABASE BUFFER POOL TRACES AT END TIME ON MONITOR LOG TAPE
****DATABASE BUFFER POOL REPORT CANCELLED****

Similarly, other summary reports are not produced.

The series of reports titled Buffer Pool Statistics do not include a VSAM Buffer Pool section unless the database in IMS.ACBLIB uses the VSAM access method. If VSAM is not used, the following message is issued:

NO VSAM BUFFER POOL TRACES ON MONITOR LOG TAPE
****VSAM BUFFER POOL REPORT CANCELLED****

If the source data used to formulate a particular IMS Monitor report, or a section of that report, has not been recorded by the IMS Monitor during the trace interval, the report contains only the headings.

Monitoring Activity in Dependent Regions

The IMS Monitor gathers timing information for every dependent region identified in the /trace command (a CCTL thread) active during the trace interval. It records the total of the elapsed times for each event, the maximum individual time encountered, and the average time.

There are three major reports that display timings. The reports and a list of their content are:

- Region Summary Report
Monitoring Activity in Dependent Regions

- Scheduling and termination
- Schedule end to first call
- Elapsed execution with separate summaries shown for:
  - DL/I calls
  - External subsystem service and command calls
  - External subsystem database access calls
  - Checkpoint processing
  - Region occupancy

• Region Wait
  - Waits during scheduling and termination
  - Waits during DL/I calls
  - Waits during external subsystem calls
  - Waits during checkpoint

• Programs by Region
  - Elapsed execution
  - Schedule end to first call

In this report, “program name” is the PSB name for the UOR.

These three reports are illustrated in Figure 127 on page 392, Figure 128 on page 393, and Figure 129 on page 394.

Activities in dependent regions are placed in five timing categories:
• Elapsed time for scheduling and termination
  The scheduling process includes many preparatory events such as block loading from an active IMS.ACBLIBA/B data set, and obtaining ownership of the PSB. The time required to terminate is the time it takes DBCTL to complete this process after receiving a request to terminate the UOR.
• Elapsed time from end of schedule to first call
  This is the time from when DBCTL completes scheduling until the time DBCTL reviews the first DL/I call. Events that occur during this time are all outside of DBCTL, either in the DRA or the CCTL.
• Program elapsed time, including all calls
  This time encompasses the major UOR processing, measured from the first DL/I call to the call that terminates a UOR.
• Elapsed time performing DL/I calls
  This time includes all DL/I calls. The time in DBCTL is recorded and summed.
## Monitoring Activity in Dependent Regions

### IMS Monitor

**Region Summary (A)**

<table>
<thead>
<tr>
<th>REGION</th>
<th>TOTAL</th>
<th>MEAN</th>
<th>MAXIMUM</th>
<th>TOTAL</th>
<th>MEAN</th>
<th>MAXIMUM</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TOTALS</strong></td>
<td>123</td>
<td>126042</td>
<td>1024</td>
<td>126042</td>
<td>1024</td>
<td><strong>TOTALS</strong></td>
<td><strong>TOTALS</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OCCURRENCES</th>
<th>ELAPSED TIME</th>
<th>NOT IN WAIT TIME (ELAPSED - IN WAIT)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Region</strong></td>
<td><strong>TOTAL</strong></td>
<td><strong>MEAN</strong></td>
</tr>
<tr>
<td><strong>TOTALS</strong></td>
<td><strong>TOTALS</strong></td>
<td><strong>TOTALS</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SCHEDULE AND TERMINATION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TOTALS</strong></td>
</tr>
</tbody>
</table>

### Schedule to First Call

<table>
<thead>
<tr>
<th>REGION</th>
<th>TOTAL</th>
<th>MEAN</th>
<th>MAXIMUM</th>
<th>TOTAL</th>
<th>MEAN</th>
<th>MAXIMUM</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TOTALS</strong></td>
<td>168</td>
<td>514286738</td>
<td>3061230</td>
<td><strong>TOTALS</strong></td>
<td><strong>TOTALS</strong></td>
<td><strong>TOTALS</strong></td>
<td><strong>TOTALS</strong></td>
</tr>
</tbody>
</table>

### Elapsed Execution

<table>
<thead>
<tr>
<th>REGION</th>
<th>TOTAL</th>
<th>MEAN</th>
<th>MAXIMUM</th>
<th>TOTAL</th>
<th>MEAN</th>
<th>MAXIMUM</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TOTALS</strong></td>
<td>173</td>
<td>1423854014</td>
<td>82303700</td>
<td><strong>TOTALS</strong></td>
<td><strong>TOTALS</strong></td>
<td><strong>TOTALS</strong></td>
<td><strong>TOTALS</strong></td>
</tr>
</tbody>
</table>

### DL/I Calls

<table>
<thead>
<tr>
<th>REGION</th>
<th>TOTAL</th>
<th>MEAN</th>
<th>MAXIMUM</th>
<th>TOTAL</th>
<th>MEAN</th>
<th>MAXIMUM</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TOTALS</strong></td>
<td>18632</td>
<td>12386905286</td>
<td>648418</td>
<td><strong>TOTALS</strong></td>
<td><strong>TOTALS</strong></td>
<td><strong>TOTALS</strong></td>
<td><strong>TOTALS</strong></td>
</tr>
</tbody>
</table>

### Idle for Intent

<table>
<thead>
<tr>
<th>CHECKPOINT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TOTALS</strong></td>
</tr>
</tbody>
</table>

### Region Occupancy

<table>
<thead>
<tr>
<th>REGION</th>
<th>TOTAL</th>
<th>MEAN</th>
<th>MAXIMUM</th>
<th>TOTAL</th>
<th>MEAN</th>
<th>MAXIMUM</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TOTALS</strong></td>
<td>100.0%</td>
<td><strong>TOTALS</strong></td>
<td><strong>TOTALS</strong></td>
<td><strong>TOTALS</strong></td>
<td><strong>TOTALS</strong></td>
<td><strong>TOTALS</strong></td>
<td><strong>TOTALS</strong></td>
</tr>
</tbody>
</table>

---

**Figure 127. Region Summary Report**
### Monitoring Activity in Dependent Regions

**Figure 128. Region Wait Report**

<table>
<thead>
<tr>
<th><strong>Region</strong></th>
<th><strong>5 Occurrences</strong></th>
<th><strong>TOTAL</strong></th>
<th><strong>FUNCTION</strong></th>
<th><strong>MODULE</strong></th>
<th><strong>NUMBER</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>TOTAL</strong></td>
<td><strong>MEAN</strong></td>
<td><strong>MAXIMUM</strong></td>
<td><strong>NUMBER</strong></td>
<td></td>
</tr>
<tr>
<td><strong>SCHEDULING + TERMINATION</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SUB-TOTAL</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**DL/I Calls**

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>181816</td>
<td>16528</td>
<td>24375</td>
<td>DO=IMMSTR2A</td>
<td>DBH</td>
</tr>
<tr>
<td>8</td>
<td>112831</td>
<td>14103</td>
<td>17846</td>
<td>DO=IMMSTR1A</td>
<td>DBH</td>
</tr>
<tr>
<td>5</td>
<td>85460</td>
<td>17092</td>
<td>33717</td>
<td>DO=IMMSTR3A</td>
<td>DBH</td>
</tr>
<tr>
<td>5</td>
<td>58420</td>
<td>11604</td>
<td>14643</td>
<td>DO=IMINDEXA</td>
<td>VBH</td>
</tr>
<tr>
<td>12</td>
<td>173866</td>
<td>14488</td>
<td>22152</td>
<td>DO=PROCNTA</td>
<td>VBH</td>
</tr>
<tr>
<td>3</td>
<td>100576</td>
<td>33525</td>
<td>68373</td>
<td>DO=IMMSTR2B</td>
<td>DBH</td>
</tr>
<tr>
<td>1</td>
<td>17921</td>
<td>17921</td>
<td>17921</td>
<td>DO=IMMSTR3B</td>
<td>DBH</td>
</tr>
<tr>
<td>1</td>
<td>17195</td>
<td>17195</td>
<td>17195</td>
<td>DO=IMMSTR1B</td>
<td>DBH</td>
</tr>
<tr>
<td>1</td>
<td>13577</td>
<td>13577</td>
<td>13577</td>
<td>DO=IMINDEXB</td>
<td>VBH</td>
</tr>
<tr>
<td>3</td>
<td>49928</td>
<td>16642</td>
<td>20396</td>
<td>DO=PROCNTB</td>
<td>VBH</td>
</tr>
<tr>
<td>4</td>
<td>10973</td>
<td>2743</td>
<td>2787</td>
<td>DO=ITEMACTB</td>
<td>DBH</td>
</tr>
<tr>
<td>2</td>
<td>37680</td>
<td>18840</td>
<td>27664</td>
<td>DO=IAINDEXB</td>
<td>VBH</td>
</tr>
<tr>
<td>49</td>
<td>1500067</td>
<td>30613</td>
<td>138284</td>
<td>DO=INVENTRA</td>
<td>DBH</td>
</tr>
<tr>
<td>23</td>
<td>345595</td>
<td>15025</td>
<td>27613</td>
<td>DO=VENDORDA</td>
<td>VBH</td>
</tr>
<tr>
<td>1</td>
<td>34292</td>
<td>34292</td>
<td>34292</td>
<td>PI=VENDORDA...1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>14612</td>
<td>14612</td>
<td>14612</td>
<td>PI=VNSINDXA...1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>69203</td>
<td>11533</td>
<td>19492</td>
<td>DO=VNSINDXA</td>
<td>VBH</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td>136</td>
<td>3132672</td>
<td>23034</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Monitoring Activity in Dependent Regions

<table>
<thead>
<tr>
<th>IMS MONITOR</th>
<th><strong><strong>PROGRAMS BY REGION</strong></strong></th>
<th>TRACE START</th>
<th>1993 130</th>
<th>5:55:15</th>
<th>TRACE STOP</th>
<th>1993 130</th>
<th>5:59:49</th>
<th>PAGE 0069</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>REGION</strong> 1</td>
<td><strong>OCCURRENCES</strong></td>
<td><strong>ELAPSED TIME</strong></td>
<td><strong>EXECUTION TIME</strong></td>
<td><strong>SCHEDULING END TO FIRST CALL</strong></td>
<td><strong>DISTRIBUTION</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong><strong>REGION</strong></strong> 1</td>
<td>1</td>
<td>29016255</td>
<td>29016255</td>
<td>15479797</td>
<td>885A,B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REGION TOTALS</td>
<td>1</td>
<td>29016255</td>
<td>29016255</td>
<td>15479797</td>
<td>885A,B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 2</td>
<td><strong>OCCURRENCES</strong></td>
<td><strong>ELAPSED TIME</strong></td>
<td><strong>EXECUTION TIME</strong></td>
<td><strong>SCHEDULING END TO FIRST CALL</strong></td>
<td><strong>DISTRIBUTION</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong><strong>REGION</strong></strong> 2</td>
<td>1</td>
<td>25220108</td>
<td>25220108</td>
<td>22376350</td>
<td>809A,B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REGION TOTALS</td>
<td>1</td>
<td>25220108</td>
<td>25220108</td>
<td>22376350</td>
<td>809A,B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 3</td>
<td><strong>OCCURRENCES</strong></td>
<td><strong>ELAPSED TIME</strong></td>
<td><strong>EXECUTION TIME</strong></td>
<td><strong>SCHEDULING END TO FIRST CALL</strong></td>
<td><strong>DISTRIBUTION</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong><strong>REGION</strong></strong> 3</td>
<td>1</td>
<td>25949670</td>
<td>25949670</td>
<td>15169488</td>
<td>893A,B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REGION TOTALS</td>
<td>1</td>
<td>25949670</td>
<td>25949670</td>
<td>15169488</td>
<td>893A,B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 4</td>
<td><strong>OCCURRENCES</strong></td>
<td><strong>ELAPSED TIME</strong></td>
<td><strong>EXECUTION TIME</strong></td>
<td><strong>SCHEDULING END TO FIRST CALL</strong></td>
<td><strong>DISTRIBUTION</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong><strong>REGION</strong></strong> 4</td>
<td>1</td>
<td>32281216</td>
<td>32281216</td>
<td>48146258</td>
<td>897A,B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REGION TOTALS</td>
<td>1</td>
<td>32281216</td>
<td>32281216</td>
<td>48146258</td>
<td>897A,B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong> 5</td>
<td><strong>OCCURRENCES</strong></td>
<td><strong>ELAPSED TIME</strong></td>
<td><strong>EXECUTION TIME</strong></td>
<td><strong>SCHEDULING END TO FIRST CALL</strong></td>
<td><strong>DISTRIBUTION</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong><strong>REGION</strong></strong> 5</td>
<td>2</td>
<td>62893103</td>
<td>40693590</td>
<td>31446551</td>
<td>1667791</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REGION TOTALS</td>
<td>6</td>
<td>27298046</td>
<td>45496743</td>
<td>1681851</td>
<td>280308</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 129. Programs-by-Region Report**

### Detecting Database Processing Intent Conflicts

The IMS Monitor records the intervals when a region is in an idle state waiting to update a database owned exclusively by another already scheduled application program.

You can see the total, maximum, and average idle times in the section IDLE FOR INTENT following the DL/I calls. The elapsed time during the unsuccessful scheduling of a program in that region is included in the summary line times for that region.

The region can fail to be scheduled even when ownership of that database is released. The number of times processing is held up by intent failure is separately tallied under the title INTENT FAILURE SUMMARY. The report is illustrated in Figure 102 on page 359. This report shows which PSBs are in conflict because of exclusive intent for a segment type and the database name in question.
Examining the Effects of Checkpoints

The checkpoint line of the Region Summary report at the end of the region 0 summary, shows the following:

- The number of times that a system checkpoint is taken during the monitor interval
- The elapsed times
- The not-wait times

Checkpoint processing can be initiated by the control program at a specified frequency determined by the number of records placed on the system log. Other checkpoints can be caused by operator commands.

The wait time experienced during checkpoints is reported at the end of the first region summary on the Region Wait report. You can detect delays for each combination of DDname and module code. Typical entries here are for the message queue data sets and the restart data set. If a wait for storage is the cause, the entry under the FUNCTION column is STG. = followed by the identification of the pool.

Measuring Region Occupancy

Region occupancy shows the ratio of elapsed time a PST spent processing UORs to the total time of the monitor interval.

Monitoring Application Program Elapsed Time

The IMS Monitor can record measurements of elapsed times for each UOR. It does this during the monitored interval while other UORs are executing concurrently. Elapsed times are calculated from the start of the first DL/I (or other) call to the end of that program. You can distinguish between time spent in application code and in DL/I processing. The event intervals are illustrated in Figure 130.

Within the elapsed time for a DL/I call, the wait time to obtain segment data is recorded separately. Also, the elapsed time from schedule to first call is separately recorded. This time covers the processing in the CCTL and the DRA.
Monitoring Application Program Elapsed Time

The elapsed times are given in the Program Summary report. Figure 131 is an example of the report. Programs are identified by their PSB name on individual lines in the report. Each line gives a summary of the activity for that PSB during the measured interval. The total number of schedules, DL/I calls, transactions dequeued, and waits for a DL/I call for I/O are given. The report line gives calculated average times for elapsed time per schedule, processor time per schedule, schedule to first DL/I call per schedule, and elapsed time per transaction. Frequencies for calls per transaction, I/O waits per DL/I call, and transactions dequeued per schedule are also given. A TOTALS line summarizes all activity for the PSBs active during the monitored interval. (The PSB DUMMY line is a reconciliation for any incomplete scheduling caused by a region being stopped during scheduling or for a program that experiences a pseudoabort.)

In this report, transaction and schedule can be interpreted as UOR.

---

**Figure 131. Program Summary Report**

To examine the detail of the call processing for each program (itemized by type of call and summarized for the monitor interval), you can use the Call Summary report. An extract from the multipage output is given in Figure 132 on page 397. The calls using an I/O PCB are given first and subtotaled. Then, the total calls, of each type, against each database PCB and each external subsystem are listed. The PSB TOTAL line marks the end of data for each program.
Monitoring Application Program Elapsed Time

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CALLS</td>
<td>LEV</td>
<td>STAT</td>
<td>PSB NAME</td>
<td>PCB NAME</td>
</tr>
<tr>
<td>PROGSC6B I/O PCB</td>
<td>ISAT ( )</td>
<td>138</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>GU ( )</td>
<td>134</td>
<td>0.99</td>
<td>2600917</td>
<td>20974615</td>
</tr>
<tr>
<td>(GU)</td>
<td>3</td>
<td>15</td>
<td>16</td>
<td>15</td>
</tr>
<tr>
<td>ASRT ( )</td>
<td>3</td>
<td>0</td>
<td>0.00</td>
<td>330</td>
</tr>
<tr>
<td>GU ( )</td>
<td>QC</td>
<td>2</td>
<td>1</td>
<td>0.50</td>
</tr>
<tr>
<td>I/O PCB SUBTOTAL</td>
<td></td>
<td>280</td>
<td>134</td>
<td>0.47</td>
</tr>
<tr>
<td>INVENTRB DLET (03)</td>
<td>IN060SUP</td>
<td>138</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>GNP (03)</td>
<td>IN060SUP</td>
<td>137</td>
<td>7</td>
<td>0.05</td>
</tr>
<tr>
<td>GU (01)</td>
<td>NO10PAR</td>
<td>138</td>
<td>1.84</td>
<td>29511</td>
</tr>
<tr>
<td>DL/I PCB SUBTOTAL</td>
<td></td>
<td>414</td>
<td>261</td>
<td>0.63</td>
</tr>
<tr>
<td>PSB TOTAL</td>
<td></td>
<td>694</td>
<td>395</td>
<td>0.56</td>
</tr>
<tr>
<td>PROGSC2A I/O PCB</td>
<td>ISAT ( )</td>
<td>118</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>GU ( )</td>
<td>114</td>
<td>2.49</td>
<td>3304809</td>
<td>21784513</td>
</tr>
<tr>
<td>(GU)</td>
<td>2</td>
<td>0</td>
<td>0.00</td>
<td>17</td>
</tr>
<tr>
<td>ASRT ( )</td>
<td>3</td>
<td>0</td>
<td>0.00</td>
<td>367</td>
</tr>
<tr>
<td>GU ( )</td>
<td>QC</td>
<td>2</td>
<td>5</td>
<td>2.50</td>
</tr>
<tr>
<td>I/O PCB SUBTOTAL</td>
<td></td>
<td>229</td>
<td>289</td>
<td>1.20</td>
</tr>
<tr>
<td>LOGVENDA REPL (03)</td>
<td>IN040SLQ</td>
<td>118</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>GNP (03)</td>
<td>IN040SLQ</td>
<td>115</td>
<td>5</td>
<td>0.04</td>
</tr>
<tr>
<td>REPL (02)</td>
<td>VN030PAR</td>
<td>826</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>GNP (02)</td>
<td>VN030PAR</td>
<td>826</td>
<td>873</td>
<td>1.05</td>
</tr>
<tr>
<td>REPL (01)</td>
<td>VN020REO</td>
<td>118</td>
<td>58</td>
<td>0.49</td>
</tr>
<tr>
<td>GU (01)</td>
<td>VN020REO</td>
<td>118</td>
<td>195</td>
<td>1.65</td>
</tr>
<tr>
<td>DL/I PCB SUBTOTAL</td>
<td></td>
<td>2124</td>
<td>1131</td>
<td>0.53</td>
</tr>
<tr>
<td>PSB TOTAL</td>
<td></td>
<td>2363</td>
<td>1420</td>
<td>0.60</td>
</tr>
<tr>
<td>PROGSC2D I/O PCB</td>
<td>ISAT ( )</td>
<td>14</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>GU ( )</td>
<td>14</td>
<td>2.57</td>
<td>22360408</td>
<td>52048566</td>
</tr>
<tr>
<td>I/O PCB SUBTOTAL</td>
<td></td>
<td>28</td>
<td>36</td>
<td>1.28</td>
</tr>
<tr>
<td>LOGVENDO REPL (03)</td>
<td>IN040SLQ</td>
<td>14</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>GNP (03)</td>
<td>IN040SLQ</td>
<td>14</td>
<td>1</td>
<td>0.07</td>
</tr>
<tr>
<td>REPL (02)</td>
<td>VN030PAR</td>
<td>98</td>
<td>0</td>
<td>0.00</td>
</tr>
</tbody>
</table>

**Figure 132. Call Summary Report**

Monitoring I/O for Application Program DL/I Calls

The IMS Monitor report shows the total number of I/O occurrences and the total time the occurrences took for each UOR during a monitored interval. The Program I/O report gives these two totals for all PSBs active during the monitored interval and includes the detailed breakdown of the I/O wait time as it was incurred by each PCB used by the program. [Figure 106 on page 364](#) shows an example of the report.

The report shows any contention experienced during application program processing. Each type of conflict and the number of times it occurred are recorded for each I/O PCB or database PCB. The report shows the total wait time, the highest wait experienced, and the average time. Subtotals are given for each PCB under a PSB, and for all PCBs under each PSB.

The DNN/FUNC column list the data set by DName. The MODULE column uses a code to indicate the source of the contention. The types of conflicts and codes are shown as follows. Any codes that appear apply to IMS only.

- **Scheduling**

<table>
<thead>
<tr>
<th>Code</th>
<th>Conflict</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLR</td>
<td>Load/read from ACBLIB</td>
</tr>
<tr>
<td>SMN</td>
<td>Virtual storage management</td>
</tr>
</tbody>
</table>
Monitoring Application Program Elapsed Time

- Database access

<table>
<thead>
<tr>
<th>Code</th>
<th>Conflict</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBH</td>
<td>OSAM I/O</td>
</tr>
<tr>
<td>DLE</td>
<td>DL/I functions</td>
</tr>
<tr>
<td>VBH</td>
<td>VSAM interface</td>
</tr>
<tr>
<td></td>
<td>(Physical segment code)</td>
</tr>
<tr>
<td></td>
<td>Program isolation</td>
</tr>
</tbody>
</table>

The I/O waits for the calls to the I/O PCB are grouped as the first entries for a PSB. For DL/I calls, the data set for which the I/O took place is indicated under the DDN/FUNC heading, and the module code tells you what type of conflict caused the wait.

Contention for the same physical segment in a database causes a wait on behalf of program isolation. This is shown in the DDN/FUNC column, on the PCB line, by the entry P1dmb, where dmb is the DMB of the physical data set. The MODULE column identifies the segment type using the physical segment code assigned by DBD generation.

When an application is accessing a database using VSAM as the access method, DL/I calls do not generally result in an I/O wait. A MODULE column entry of VBH indicates that interface to VSAM occurred and there was an I/O wait.

A seemingly unrelated entry can occur under the DDN/FUNC column for a database PCB. An example is a retrieval call to a database (DB-A) that causes a buffer to be purged in order to make room for that retrieved data. If the buffer contents included data belonging to another database (DB-B), the I/O entry in the report shows the DDname for DB-B as being in conflict for PCB access to DB-A.

Transaction Queuing Report

In the Transaction Queuing Report in Figure 133 on page 399, a list of transactions is shown for DBCTL. Each transaction name is an 8-byte transaction ID specified by the CCTL on the schedule request or the CCTL ID. A transaction ID from CICS, when used as the transaction manager, is composed of a 4-byte CICS transaction name plus a 4-byte CICS identifier. If the CCTL does not specify the transaction ID, DBCTL takes the CCTL region ID obtained at connection time as the default. In this report for DBCTL, the transaction NUMBER DEQUEUED is the number of schedules, and the ON QUEUE WHEN SCHEDULED is always zero, because the IMS message queues are not involved.
Database Monitor

**Figure 133. Transaction Queuing Report**

**Monitoring Database Buffers**

One of the key resources in an online system is the database buffer pool. The efficiency of DL/I call service depends on the presence of the required database logical record in the buffer, so that segment retrieval does not require additional I/O.

This is especially true for HOLD calls with intervening database calls prior to a replace call. You can assess the general efficiency of the pool management using the Database Buffer Pool report shown in Figure 134 on page 400. The event counts on this report are not specific to a particular database or program but represent the pressure for use of the database pool.

**Related Reading:** Refer to [IMS Version 8: Utilities Reference: Database and Transaction Manager](#) for more information on the Database Buffer Pool reports.
Monitoring Database Buffers

**Database Buffer Pool Statistics**

<table>
<thead>
<tr>
<th></th>
<th>FIX PREFIX/BUFFERS</th>
<th>Y/Y</th>
<th>SUBPOOL ID</th>
<th>004X</th>
<th>SUBPOOL BUFFER SIZE</th>
<th>4096</th>
<th>TOTAL BUFFERS IN SUBPOOL</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>START TRACE</td>
<td>END TRACE</td>
<td>DIFFERENCE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUMBER OF LOCATE-TYPE CALLS</td>
<td>407636</td>
<td>429673</td>
<td>389157</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUMBER OF REQUESTS TO CREATE NEW BLOCKS</td>
<td>1</td>
<td>7</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUMBER OF BUFFER ALTER CALLS</td>
<td>75006</td>
<td>819359</td>
<td>744353</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUMBER OF PURGE CALLS</td>
<td>9137</td>
<td>93801</td>
<td>24644</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUMBER OF LOCATE-TYPE CALLS, DATA ALREADY IN OSAM POOL</td>
<td>313696</td>
<td>3371264</td>
<td>3000368</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUMBER OF BUFFERS SEARCHED BY ALL LOCATE-TYPE CALLS</td>
<td>453364</td>
<td>4779327</td>
<td>4325963</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUMBER OF READ I/O REQUESTS</td>
<td>86881</td>
<td>904487</td>
<td>817606</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUMBER OF SINGLE BLOCK WRITES BY BUFFER STEAL ROUTINE</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUMBER OF BLOCKS WRITTEN BY PURGE</td>
<td>32629</td>
<td>360434</td>
<td>327805</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUMBER OF LOCATE CALLS WAITED DUE TO BUSY ID</td>
<td>281</td>
<td>3173</td>
<td>2982</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUMBER OF LOCATE CALLS WAITED DUE TO BUFFER BUSY WRT</td>
<td>6</td>
<td>180</td>
<td>174</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUMBER OF LOCATE CALLS WAITED DUE TO BUFFER BUSY READ</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUMBER OF BUFFER STEAL/PURGE WAITED FOR OWNERSHIP RLSE</td>
<td>43</td>
<td>483</td>
<td>440</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUMBER OF BUFFER STEAL REQUESTS WAITED FOR BUFFERS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTAL NUMBER OF I/O ERRORS FOR THIS SUBPOOL</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUMBER OF BUFFERS LOCKED DUE TO WRITE ERRORS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUOTIENT : TOTAL NUMBER OF OSAM READS + OSAM WRITES</td>
<td>7.02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Figure 134. Database Buffer Pool Report**

If any of your databases use VSAM as access method, the IMS Monitor produces a series of reports headed `VSAM BUFFER POOL`, one for each subpool. **Figure 135** shows one of these reports.

---

**Figure 135. VSAM Buffer Pool Report**

---

400 Utilities Reference: System
IMS Internal Resource Usage

There are several summary reports that you can use to examine the level of internal contention for resources. The following sections give a brief description of these reports.

Pool Space Failure

The Pool Space Failure Summary report gives the number of times (in each region) a given amount of storage was unavailable. It shows the number of bytes, the identification of the pool, and the number of occurrences when storage was unavailable. You can use this summary to determine if you need to increase the buffer pool allocation, either by a system definition change or by overriding the number of buffers in the EXEC statements in the JCL.

The format of the report is shown in Figure 136.

\[
\begin{array}{ccc}
\text{POOL ID} & \text{BYTES REQ.} & \text{OCCURRENCES} \\
\text{DLMP} & 8888 & 1 \\
\text{DLDP} & 7777 & 1 \\
\text{TOTAL} & & 2 \\
\end{array}
\]

*Figure 136. Pool Space Failure Report*

Programs Experiencing Deadlock

The Deadlock Event Summary report records each time a pair of programs reaches a deadlock over ownership of a segment in a given database data set. Each line in the report shows the two PSBs involved and indicates which is given processing right-of-way (REQ-ING PSB) and which has to reprocess after dynamic backout has occurred (LOSING PSB). The report is illustrated in Figure 137.

\[
\begin{array}{cccc}
\text{REQ-ING PSB} & \text{LOSING PSB} & \text{DMNAME} & \text{OCCURRENCES} \\
\text{PSBNAME1} & \text{TPPSBRE3} & \text{DBASEBAL} & 1 \\
\text{TOTAL} & & & 1 \\
\end{array}
\]

*Figure 137. Deadlock Event Summary Report*

IMS Latch Conflict

The basic serialization of the task processing in IMS is controlled by ownership of an IMS latch. When different programs are executing, they compete for the ownership. If they wait for the resource, the one possessing the latch has to post the other ITASK waiting for it. Use the Latch Conflict Statistics report to judge the level of contention for a resource.

The different types of latches and the counters that exhibit the level of contention are given in the Latch Conflict Statistics report. Figure 138 on page 402 is an example of this report. The entries are organized according to the latch names.

For the latch names and abbreviations of the different types of resources being serialized see "IMS Latch Conflict" on page 373.
When a system checkpoint is taken during the time the monitor is active, latch conflict statistics are reset to zero, thus corrupting the values presented in this report. If this situation exists, the following message will be inserted at the top of the report:

```
***** A CHECKPOINT OCCURRED DURING MONITOR RUN *****
***** LATCH CONFLICT STATISTICS ARE INVALID *****
***** SEE UTILITIES REFERENCE MANUAL *****
```

However, if the master terminal operator issues the \CHECKPOINT command with the STATISTICS keyword parameter, latch conflict statistics are reset to zero, but the IMS monitor is not notified. Therefore, DFSUTR20 cannot detect that the statistics have been corrupted and will not issue this message.

**Recommendation:** Do not issue statistics checkpoints while the monitor is running.

<table>
<thead>
<tr>
<th>LATCH NAMES</th>
<th>COUNT FIELD</th>
<th>AT START</th>
<th>AT END</th>
<th>DIFF.</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGL</td>
<td>CONTENTIONS</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SMGT</td>
<td>CONTENTIONS</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>XCNQ</td>
<td>CONTENTIONS</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ACTL</td>
<td>CONTENTIONS</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CBTS</td>
<td>CONTENTIONS</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DBLK</td>
<td>CONTENTIONS</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

*Figure 138. Latch Conflict Statistics Report*

### Using Frequency Distributions from IMS Monitor Output

The reports derived from the IMS Monitor data records contain many summary lines where the mean time is given. If you are interested in the distribution of those timed events, rather than just average and maximum times, you can request the Report Print utility to individually record the events in a frequency distribution across a range of intervals. Some distributions are not time dependent, such as those for transaction queue loads or transmitted block sizes.

### How to Get a Frequency Distribution Output

To request the IMS Monitor Report Print utility to gather distribution data, you must include a **DIS** input control statement. This causes all report items with an entry under a column headed **MEAN** to have a corresponding frequency distribution as part of the Distribution Appendix. Also, each report line includes an identifying reference number under the column headed **DISTRIBUTION NUMBER** so that you can locate the distribution data in the appendix, flagged by that same number.

The following tables show the major IMS Monitor reports and the type of frequency distributions generated for each report. Each type results in several distributions, depending on how many entries are in each section of the report. For each type of frequency distribution, the data is cumulated in suitable intervals or ranges. The set
of ranges used for each type is given an identifier, shown in the ID column.

Table 33 shows the report distributions sorted by Region Summary.

Table 33. Report Distributions by Region Summary

<table>
<thead>
<tr>
<th>Report Name</th>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduling and Termination</td>
<td>D1</td>
<td>Elapsed time</td>
</tr>
<tr>
<td></td>
<td>D2</td>
<td>Not wait time</td>
</tr>
<tr>
<td>Schedule end to 1st DL/I call</td>
<td>D3</td>
<td>N/A</td>
</tr>
<tr>
<td>Elapsed execution time</td>
<td>D4</td>
<td>N/A</td>
</tr>
<tr>
<td>DL/I calls</td>
<td>D5</td>
<td>Elapsed time</td>
</tr>
<tr>
<td></td>
<td>D6</td>
<td>Not wait time</td>
</tr>
<tr>
<td>External Subsystem calls</td>
<td>D43</td>
<td>Elapsed time</td>
</tr>
<tr>
<td>Waits per DL/I call</td>
<td>D7</td>
<td>N/A</td>
</tr>
<tr>
<td>Idle for intent</td>
<td>D8</td>
<td>N/A</td>
</tr>
<tr>
<td>Checkpoint</td>
<td>D20</td>
<td>Elapsed time</td>
</tr>
<tr>
<td></td>
<td>D21</td>
<td>Not wait time</td>
</tr>
</tbody>
</table>

Table 34 shows the report distributions sorted by Programs Region.

Table 34. Report Distributions by Programs Region

<table>
<thead>
<tr>
<th>Report Name</th>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elapsed execution time</td>
<td>D30</td>
<td>N/A</td>
</tr>
<tr>
<td>Schedule and to 1st DL/I call</td>
<td>D31</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 35 shows the report distributions by Program Summary.

Table 35. Report Distributions by Program Summary

<table>
<thead>
<tr>
<th>Report Name</th>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor time per schedule</td>
<td>D15</td>
<td>N/A</td>
</tr>
<tr>
<td>Transactions dequeued per schedule</td>
<td>D14</td>
<td>N/A</td>
</tr>
<tr>
<td>Elapsed time per schedule</td>
<td>D9</td>
<td>N/A</td>
</tr>
<tr>
<td>Schedule end to 1st DL/I call</td>
<td>D10</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 36 shows the report distributions sorted by Call Summary.

Table 36. Report Distributions by Call Summary

<table>
<thead>
<tr>
<th>Report Name</th>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSB waits per DL/I call</td>
<td>D13</td>
<td>N/A</td>
</tr>
<tr>
<td>PSB waits per external subsystem call</td>
<td>D44</td>
<td>N/A</td>
</tr>
<tr>
<td>PSB elapsed time per call</td>
<td>D11</td>
<td>N/A</td>
</tr>
<tr>
<td>PSB not wait time per call</td>
<td>D12</td>
<td>N/A</td>
</tr>
<tr>
<td>PSB external subsystem calls</td>
<td>D45</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>N/A</td>
<td>Elapsed time</td>
</tr>
</tbody>
</table>
Table 37 lists some distributions derived from buffer pool statistics for wait times.

### Table 37. Wait Time Distributions

<table>
<thead>
<tr>
<th>Function</th>
<th>ID</th>
<th>Module Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage</td>
<td>D22</td>
<td>SMN</td>
</tr>
<tr>
<td>OSAM I/O</td>
<td>D23</td>
<td>DBH</td>
</tr>
<tr>
<td>VSAM I/O</td>
<td>D24</td>
<td>VBH</td>
</tr>
<tr>
<td>Block loader I/O</td>
<td>D27</td>
<td>BLR</td>
</tr>
<tr>
<td>HSAM I/O</td>
<td>D34</td>
<td>DIE</td>
</tr>
<tr>
<td>PI enqueue</td>
<td>D40</td>
<td>None</td>
</tr>
</tbody>
</table>

### How Frequency Distribution Ranges Are Defined

A set of ten intervals is defined for each summary line and the occurrences falling in each interval are cumulated. The interval ranges are preset with default end points. For example, the end points, for DL/I call elapsed time are: 0, 1000, 2000, 4000, 8000, 16000, 32000, 64000, 128000, 256000, INF (all times are in milliseconds). The default end points are chosen so that they are suitable to the event. The lower limit of the first interval always defaults to zero, and the upper limit of the tenth interval is infinity (INF).

Although several types of distribution can use the same set of end points, each type is assigned a distribution identifier. You can use this to redefine the end points. To override the default end points include an input control statement to the Report Print utility. The statement specifies the type of distribution identifier and gives the desired end point values. For example, the DL/I call elapsed time end points could be respecified by:

```
D5 0,500,1000,1500,2000,4000,,100000,500000
```

The values of the unspecified end points remain at their default values of 32000 and 64000 as does the last (INF).

Figure 139 on page 405, which is a sample page taken from a Distribution Appendix, shows how individual distributions are numbered and how ranges vary with the type of distribution. The lines are arranged in pairs, with the second one recording the cumulated counts.
### Default Values of Distribution Definitions

Using an identifier provided in the frequency distribution tables (Table 33 on page 404) you can determine the default end points for the distribution by locating it in the following list:

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Default End Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1, D2, D3, D4, D5, D6, D7, D8, D9, D10, D11, D12, D13, D14, D15, D16, D17, D18, D19, D20, D21, D22, D23, D24, D25, D26, D27</td>
<td>0, 0, 1000, 2000, 4000, 8000, 16000, 32000, 64000, 128000, 256000, INF</td>
</tr>
<tr>
<td>D28, D29, D30, D31, D32, D33, D34, D35</td>
<td>0, 0, 1000, 2000, 4000, 8000, 16000, 32000, 64000, 128000, 256000, INF</td>
</tr>
<tr>
<td>D36, D37, D38, D39, D40, D41, D42, D43, D44, D45</td>
<td>0, 0, 1000, 2000, 4000, 8000, 16000, 32000, 64000, 128000, 256000, INF</td>
</tr>
</tbody>
</table>

---

**Table 33** on pages 403 through 406, and the Wait Time Distribution Table (Table 37) on page 404, you can determine the default end points for the distribution by locating it in the following list:

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Default End Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1, D2, D3, D4, D5, D6, D7, D8, D9, D10, D11, D12, D13, D14, D15, D16, D17, D18, D19, D20, D21, D22, D23, D24, D25, D26, D27</td>
<td>0, 0, 1000, 2000, 4000, 8000, 16000, 32000, 64000, 128000, 256000, INF</td>
</tr>
<tr>
<td>D28, D29, D30, D31, D32, D33, D34, D35</td>
<td>0, 0, 1000, 2000, 4000, 8000, 16000, 32000, 64000, 128000, 256000, INF</td>
</tr>
<tr>
<td>D36, D37, D38, D39, D40, D41, D42, D43, D44, D45</td>
<td>0, 0, 1000, 2000, 4000, 8000, 16000, 32000, 64000, 128000, 256000, INF</td>
</tr>
</tbody>
</table>
Using Frequency Distributions

| D36, D37 | 0, 10, 20, 40, 80, 100, 200, 400, 800, 1000, INF |
| D38      | 0, 1000, 10000, 100000, 200000, 500000, 800000, 1000000, 1500000, 2000000, INF |
| D39      | 0, 1000, 5000, 10000, 50000, 100000, 500000, 1000000, 5000000, 10000000, INF |

Interpreting Distribution Appendix Output

You can use the detailed output in the Distribution Appendix when you suspect an unusual combination of events was reported in a report summary line. Usually, the average and maximum times or counts are sufficient to highlight a resource usage problem. However, if you suspect the mean value to be masking an unusual distribution you can draw on the detail contained in the IMS Monitor output records.
Chapter 19. Interpreting IMS Monitor Reports for DCCTL

DCCTL is a transaction management subsystem that has no database components. With the external subsystem (ESS) attach facility, it provides the transaction management capability for non-IMS database subsystems.

This chapter describes:
- The events that the IMS Monitor collects
- The content of the reports produced by the IMS Monitor Report Print Program in a DCCTL environment

DCCTL does not change the format or usage of the IMS Monitor reports. There are reports, and fields within reports, that contain information specific to databases, and these are not applicable to the DCCTL environment. Reports that do not apply to DCCTL appear as a heading without data, or are not produced. The reports that do not apply to DCCTL include:
- Database Buffer Pool report
- VSAM Buffer Pool (DB) report
- Call Summary (DB) report
- Program I/O (DB) report

For a detailed look at the events, system activities, and use of storage areas (buffer pool or data set) for which timings are gathered by the IMS Monitor, see Table 21 on page 347.

In this Chapter:
- “IMS Monitor Trace Event Intervals”
- “Overview of IMS Monitor Reports” on page 408
- “Documenting the Monitoring Run” on page 409
- “Monitoring Activity in Dependent Regions” on page 412
- “Monitoring Application Program Elapsed Time” on page 417
- “Monitoring I/O for Application Program DL/I Calls” on page 419
- “Monitoring MFS Activity” on page 423
- “Monitoring Message Queue Handling” on page 424
- “Monitoring Line Activity” on page 426
- “Monitoring Message Handling Efficiency” on page 427
- “IMS Internal Resource Usage” on page 428
- “Using Frequency Distributions from IMS Monitor Output” on page 429
- “Interpreting IMS Monitor MSC Reports” on page 434
- “Extracting Multiple System Transaction Statistics” on page 437

IMS Monitor Trace Event Intervals

The IMS Monitor trace interval is defined by the master terminal operator’s use of the /TRACE command between the start and stop command entries. The online IMS events are recorded in IMS Monitor records placed in the IMSMON data set. The event timings are related to dependent region activity. Figure 140 on page 408 shows the boundaries of the timed event intervals.

The Monitor trace interval includes the following intervals:
**Overview of IMS Monitor Reports**

A list of reports available from data collected by the IMS Monitor, together with the principal performance data they contain, is shown in Table 38. The reports marked “MSC” in the list are only produced when MSC is active. The MSC reports are discussed in “Interpreting IMS Monitor MSC Reports” on page 434.

**Sequence of Report Output**

The order of the reports listed in Table 38 matches the sequence of the output from the IMS Monitor Report Print Program. The duration of a monitoring snapshot might not include certain events necessary for an individual report, in which case only report headings or partial data are produced.

**Summary of IMS Monitor Reports in Output Sequence**

<table>
<thead>
<tr>
<th>Report Name</th>
<th>Principal Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Configuration</td>
<td>Monitor run documentation</td>
</tr>
<tr>
<td>Message Queue Pool</td>
<td>Buffering and message I/O per transaction</td>
</tr>
<tr>
<td>Message Format Buffer Pool</td>
<td>Count of I/Os</td>
</tr>
</tbody>
</table>
Table 38. Output Sequence and Information from IMS Monitor Reports (continued)

<table>
<thead>
<tr>
<th>Report Name</th>
<th>Principal Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latch Conflict Statistics</td>
<td>IMS internal processing</td>
</tr>
<tr>
<td>General Wait Time Events</td>
<td>Wait times for SNAPQ</td>
</tr>
<tr>
<td>Region and Jobname</td>
<td>Monitor run documentation</td>
</tr>
<tr>
<td>Region Summary</td>
<td>Elapsed times and count of DL/I calls (DC)</td>
</tr>
<tr>
<td>Region Wait</td>
<td>Wait times</td>
</tr>
<tr>
<td>Programs by Region</td>
<td>Elapsed times for region usage</td>
</tr>
<tr>
<td>Program Summary</td>
<td>Overall program statistics</td>
</tr>
<tr>
<td>Communication Summary</td>
<td>Elapsed times for lines</td>
</tr>
<tr>
<td>Communication Wait</td>
<td>Wait times by line</td>
</tr>
<tr>
<td>Line Functions</td>
<td>Count and size of blocks transmitted</td>
</tr>
<tr>
<td>MSC Traffic (MSC)</td>
<td>Count and routing of transactions</td>
</tr>
<tr>
<td>MSC Summaries (MSC)</td>
<td>Count of transactions by destination</td>
</tr>
<tr>
<td>MSC Queuing Summary (MSC)</td>
<td>Count and queuing time by link</td>
</tr>
<tr>
<td>Transaction Queuing</td>
<td>Queue loading statistics</td>
</tr>
<tr>
<td>Reports</td>
<td>Count of space failures and deadlocks</td>
</tr>
<tr>
<td>Run Profile</td>
<td>Monitor run documentation</td>
</tr>
<tr>
<td>Distribution Appendix</td>
<td>Event frequency distributions</td>
</tr>
</tbody>
</table>

**Units of Measure in IMS Monitor Reports**

The majority of the data items in IMS Monitor reports are elapsed times. These are normally expressed in microseconds. An entry of 1876534 is 1.876534 seconds or 1876 milliseconds. Any times that do not follow this convention show the unit of measure on the report.

You can also find counts of events under the heading OCCURRENCES, and figures that represent the number of bytes.

**Documenting the Monitoring Run**

For each trace interval there are several general reports or overall summaries of the processing that took place. You can use these reports as part of your IMS Monitor run documentation.

It is important to record as accurately as possible the conditions under which the trace was taken. Your documentation can include system status information obtained by the /DISPLAY command several times before and after the trace, an expected profile of the application program activity, and any desired processing events. The trace interval should represent typical processing loads and not be a biased or inadequate historical record.

**Adding to the System Configuration Report Data**

The first general report titled SYSTEM CONFIGURATION is found under the page heading BUFFER POOL STATISTICS. It shows the modification level of the IMS and MVS systems. You can add a list of IMS APARs applied and include the service
Documenting the Monitoring Run

levels of the application programs, especially if the latter are not permanent programs or are part of a staged implementation. The system configuration output is illustrated in [Figure 141].

Recording the Monitor Trace Interval

The heading of most IMS Monitor reports carries the trace start and stop times. It is shown in the format YEAR DAY (Julian) HH:MM:SS. The overall length of the trace interval is given in seconds under the heading TRACE ELAPSED TIME IN SECONDS. The following line shows how many trace records were placed on the IMS.MON data set. An example of the monitor trace interval recording is shown in [Figure 141].

SYSTEM CONFIGURATION

SYSTEM CONFIGURATION :
IMS VERSION : 4
RELEASE LEVEL :
MODIFICATION NUMBER :

Figure 141. IMS Monitor System Configuration Report and Trace Interval

Completing the Monitor Run Profile

A compact set of processing ratios is found at the end of the Run Profile report. The statistics summarize, for the monitor interval, the transaction throughput and the degree of DL/I and I/O activity. An example of the report is shown in [Figure 142 on page 411]. In a DCCTL environment, DL/I activity is restricted to data communications calls and calls to GSAM databases. Database calls to other types of DL/I databases are not supported in DCCTL.

The lower part of the Run Profile report shows several ratios:

- Program elapsed time to DL/I elapsed time for each region
- DL/I elapsed time to wait time during DL/I processing
- Program elapsed time to other subsystem call elapsed time
- DL/I elapsed time to other subsystem call elapsed time

Each dependent region is identified by a sequence number, starting at region 1.
You can match the regions to the MVS job name using the Region and Jobname report. The job names correspond to the step names on the EXEC statements of all the dependent regions started by the operator before the trace was started. The region job names are included on the monitor output page with the heading GENERAL REPORTS, as illustrated in Figure 152 on page 425.

There are some generalized processing ratios that are given at the end of several buffer pool statistics reports. You can include them in the documented profile of the trace interval. These are not specific to one application or system resource but can be used as indicators of variation across a series of monitor runs. In DCCTL, the ratios are:

- All waits divided by the total number of transactions
  This value can be found on the Message Queue Pool Report in Figure 151 on page 424. This ratio indicates on a per transaction basis the physical I/O activity required to handle the message queuing function.
- The total prefetch I/Os + immediate fetch I/Os + directory I/Os divided by the total number of transactions
  This value also appears on Figure 150 on page 424. This ratio indicates on a per transaction basis the physical I/O activity required to handle the MFS function during the trace period.

Verifying IMS Monitor Report Occurrences

When you examine the output from the IMS Monitor Report Print program, do not assume that the presence of a report heading implies that appropriate data is listed. System definition options and utility control statements affect the content of the output as follows:

- The output does not include a Call Summary report unless a control statement specifies DLI.
Documenting the Monitoring Run

- The output does not include a set of Distribution reports unless a control statement specifies DIS or DISTRIBUTION. The column headed DISTRIBUTION NUMBER that occurs on many of the reports contains cross-references to items included in the Distribution reports.
- The output consists of just a Call Summary report if a control statement specifies ONLY DLI.

Because many of the summary reports require system status to calculate the difference between start and end values, and this status is obtained during /TRACE SET OFF processing, the IMS Monitor execution must end before termination of the IMS control region. If the trace was not stopped properly, the following message is issued:

NO QUEUE BUFFER POOL TRACES AT END TIME ON MONITOR LOG TAPE
****QUEUE BUFFER POOL REPORT CANCELLED****

Similarly, other summary reports are not produced.

The section MESSAGE FORMAT BUFFER POOL is included only if your system definition specifies devices using Message Format Service.

If the source data used to formulate a particular IMS Monitor report, or a section of that report, has not been recorded by the IMS Monitor during the trace interval, the report contains only the headings.

Monitoring Activity in Dependent Regions

The IMS Monitor gathers timing information for every dependent region identified in the /trace command active during the trace interval. It records the total of the elapsed times for each event, the time for the longest event encountered, and the average time for all recorded events.

There are three major reports that display timings. The reports and a list of their content are:

- **Region Summary Report**
  - Scheduling and termination
  - Schedule end to first call
  - Elapsed execution with separate summaries shown for:
    - DL/I calls
    - External subsystem service and command calls
    - External subsystem database access calls
    - Checkpoint processing
    - Region occupancy
- **Region Wait**
  - Waits during scheduling and termination
  - Waits during DL/I calls
  - Waits during external subsystem calls
  - Waits during checkpoint
- **Programs by Region**
  - Elapsed execution
  - Schedule end to first call
Activities in dependent regions are placed in five timing categories:

- **Elapsed time for scheduling and termination**
  The scheduling process includes many preparatory events such as block loading from an active IMS.ACBLIBA/B data set and obtaining ownership of the PSB. The time required to terminate the region activity after the application program ends is also included.

- **Elapsed time from end of schedule to first call**
  This time is reserved for application program initialization and housekeeping prior to an initial call (to the message queue, or an external subsystem) that marks the beginning of control program services. It is a measure of processing that is not repeated when multiple transactions are processed in a single scheduling.

- **Program elapsed time, including all calls**
  This time encompasses the major application program processing and is measured from the first call to the return to or exit from the program.

- **Elapsed time performing DL/I calls**
  This time includes all DL/I calls. Each DL/I call event is measured from the time of the call to the return to the application program.

- **Elapsed time performing external subsystem calls**
  This time includes all external subsystem calls. Each external subsystem event is measured from the time of the call to the return to IMS.
## Monitoring Activity in Dependent Regions

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scheduling and Termination</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong></td>
<td><strong>TOTALS</strong></td>
<td><strong>OCCURRENCES</strong></td>
<td><strong>Elapsed Time</strong></td>
<td><strong>Not IWait Time (Elapsed-IWait)</strong></td>
<td><strong>DISTRIBUTION</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION</strong></td>
<td><strong>TOTAL</strong></td>
<td><strong>MEAN</strong></td>
<td><strong>MAXIMUM</strong></td>
<td><strong>TOTAL</strong></td>
<td><strong>MEAN</strong></td>
<td><strong>MAXIMUM</strong></td>
<td><strong>NUMBER</strong></td>
</tr>
<tr>
<td><strong>REGION</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION 1</strong></td>
<td>123</td>
<td>126042</td>
<td>1024</td>
<td>126042</td>
<td>1024</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION 2</strong></td>
<td>1</td>
<td>15479797</td>
<td>15479797</td>
<td>15479797</td>
<td>15479797</td>
<td>15479797</td>
<td>15479797</td>
</tr>
<tr>
<td><strong>REGION 3</strong></td>
<td>1</td>
<td>22376350</td>
<td>22376350</td>
<td>22376350</td>
<td>22376350</td>
<td>22376350</td>
<td>22376350</td>
</tr>
<tr>
<td><strong>REGION 4</strong></td>
<td>1</td>
<td>15169488</td>
<td>15169488</td>
<td>15169488</td>
<td>15169488</td>
<td>15169488</td>
<td>15169488</td>
</tr>
<tr>
<td><strong>REGION 5</strong></td>
<td>1</td>
<td>48146258</td>
<td>48146258</td>
<td>48146258</td>
<td>48146258</td>
<td>48146258</td>
<td>48146258</td>
</tr>
<tr>
<td><strong>REGION 6</strong></td>
<td>1</td>
<td>795351</td>
<td>795351</td>
<td>795351</td>
<td>795351</td>
<td>795351</td>
<td>795351</td>
</tr>
<tr>
<td><strong>REGION 7</strong></td>
<td>4</td>
<td>2960425</td>
<td>740106</td>
<td>2951746</td>
<td>442</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION 8</strong></td>
<td>1</td>
<td>15713464</td>
<td>15713464</td>
<td>15713464</td>
<td>15713464</td>
<td>15713464</td>
<td>15713464</td>
</tr>
<tr>
<td><strong>TOTALS</strong></td>
<td>168</td>
<td>514286738</td>
<td>3061230</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Elapsed Execution</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION 1</strong></td>
<td>1</td>
<td>1</td>
<td>290146255</td>
<td>290146255</td>
<td>290146255</td>
<td>290146255</td>
<td>290146255</td>
</tr>
<tr>
<td><strong>REGION 2</strong></td>
<td>2</td>
<td>1</td>
<td>252290108</td>
<td>252290108</td>
<td>252290108</td>
<td>252290108</td>
<td>252290108</td>
</tr>
<tr>
<td><strong>REGION 3</strong></td>
<td>3</td>
<td>1</td>
<td>259496970</td>
<td>259496970</td>
<td>259496970</td>
<td>259496970</td>
<td>259496970</td>
</tr>
<tr>
<td><strong>REGION 4</strong></td>
<td>4</td>
<td>1</td>
<td>32281716</td>
<td>32281716</td>
<td>32281716</td>
<td>32281716</td>
<td>32281716</td>
</tr>
<tr>
<td><strong>REGION 5</strong></td>
<td>48</td>
<td>1</td>
<td>273871107</td>
<td>273871107</td>
<td>273871107</td>
<td>273871107</td>
<td>273871107</td>
</tr>
<tr>
<td><strong>REGION 6</strong></td>
<td>49</td>
<td>4</td>
<td>271703421</td>
<td>67925855</td>
<td>155176058</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td><strong>REGION 7</strong></td>
<td>50</td>
<td>1</td>
<td>290379922</td>
<td>290379922</td>
<td>290379922</td>
<td>290379922</td>
<td>290379922</td>
</tr>
<tr>
<td><strong>TOTALS</strong></td>
<td>173</td>
<td>14238540145</td>
<td>82303700</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>DL/I Calls</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION 1</strong></td>
<td>1</td>
<td>60</td>
<td>264626241</td>
<td>264626241</td>
<td>264626241</td>
<td>264626241</td>
<td>264626241</td>
</tr>
<tr>
<td><strong>REGION 2</strong></td>
<td>223</td>
<td>2230505269</td>
<td>2230505269</td>
<td>2230505269</td>
<td>2230505269</td>
<td>2230505269</td>
<td>2230505269</td>
</tr>
<tr>
<td><strong>REGION 3</strong></td>
<td>29</td>
<td>257704383</td>
<td>257704383</td>
<td>257704383</td>
<td>257704383</td>
<td>257704383</td>
<td>257704383</td>
</tr>
<tr>
<td><strong>REGION 4</strong></td>
<td>289</td>
<td>313733547</td>
<td>313733547</td>
<td>313733547</td>
<td>313733547</td>
<td>313733547</td>
<td>313733547</td>
</tr>
<tr>
<td><strong>REGION 5</strong></td>
<td>36</td>
<td>242591451</td>
<td>6738651</td>
<td>242591451</td>
<td>0.80</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION 6</strong></td>
<td>36</td>
<td>242591451</td>
<td>6738651</td>
<td>242591451</td>
<td>0.80</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION 7</strong></td>
<td>50</td>
<td>242591451</td>
<td>6738651</td>
<td>242591451</td>
<td>0.80</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>TOTALS</strong></td>
<td>18632</td>
<td>12386905286</td>
<td>648018</td>
<td>12024562411</td>
<td>648018</td>
<td>0.97</td>
<td></td>
</tr>
<tr>
<td><strong>IWT/Call(C)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Region Summary</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Region Summary</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Cheekpoint</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Region Occupancy</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION 1</strong></td>
<td>100.0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION 2</strong></td>
<td>100.0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION 3</strong></td>
<td>100.0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION 4</strong></td>
<td>100.0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION 5</strong></td>
<td>100.0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION 6</strong></td>
<td>100.0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>REGION 7</strong></td>
<td>100.0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Figure 143. Region Summary Report*
## IMS Monitor Activity in Dependent Regions

**Figure 144. Region Wait Report**

<table>
<thead>
<tr>
<th><strong>Region</strong></th>
<th><strong>OCCURRENCES</strong></th>
<th><strong>TOTAL</strong></th>
<th><strong>MEAN</strong></th>
<th><strong>MAXIMUM</strong></th>
<th><strong>FUNCTION</strong></th>
<th><strong>MODULE</strong></th>
<th><strong>NUMBER</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEDULING</td>
<td></td>
<td>DM 0023</td>
<td>14 181816</td>
<td>16522 24375</td>
<td>IMSTR2A(DBH)</td>
<td></td>
<td>117</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 112831</td>
<td>14 17092</td>
<td>22463 33717</td>
<td>IMSTR3A(DBH)</td>
<td></td>
<td>119</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 58420</td>
<td>111680  14463</td>
<td>IMINDEXA(VBH)</td>
<td></td>
<td>120</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>11 173866</td>
<td>14488   22152</td>
<td>PROCNTA(VBH)</td>
<td></td>
<td>121</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 100576</td>
<td>33525   68373</td>
<td>IMSTR2B(DBH)</td>
<td></td>
<td>428</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 17921</td>
<td>17921   17921</td>
<td>IMSTR3B(DBH)</td>
<td></td>
<td>429</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 17195</td>
<td>17195   17195</td>
<td>IMSTR1B(DBH)</td>
<td></td>
<td>430</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 13577</td>
<td>13577   13577</td>
<td>IMINDEXB(VBH)</td>
<td></td>
<td>431</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 49928</td>
<td>16642   20396</td>
<td>PRODCTA(VBH)</td>
<td></td>
<td>432</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 10973</td>
<td>2743    2787</td>
<td>IMACTA(DBH)</td>
<td></td>
<td>453</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 37680</td>
<td>18840   27664</td>
<td>IAINDEXB(VBH)</td>
<td></td>
<td>454</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>49 1500067</td>
<td>30613  138284</td>
<td>INVENTRA(DBH)</td>
<td></td>
<td>472</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>23 345595</td>
<td>15025   27613</td>
<td>VENDORA(VBH)</td>
<td></td>
<td>473</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 342952</td>
<td>342952  342952</td>
<td>VNSINDX..1</td>
<td></td>
<td>499</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 14612</td>
<td>14612   14612</td>
<td>VNSINDX..1</td>
<td></td>
<td>499</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 69203</td>
<td>11533   19492</td>
<td>VNSINDX(VBH)</td>
<td></td>
<td>500</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td>136 313267</td>
<td>23034</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Figure 145. Programs-by-Region Report

### Examining the Effects of Checkpoints

The checkpoint line of the Region Summary report at the end of the region 0 summary shows the following:

- The number of system checkpoint taken during the monitor interval
- The elapsed times
- The not-wait times

Checkpoint processing can be initiated by the control program at a specified frequency determined by the number of records placed on the system log. Other checkpoints can be caused by operator commands.

The wait time experienced during checkpoints is reported at the end of the first region summary on the Region Wait report. You can detect delays for each combination of DD name and module code. Typical entries here are for the
message queue data sets and the restart data set. If an wait for storage is the cause, the entry under the FUNCTION column is STG.=, followed by the identification of the pool.

**Measuring Region Occupancy**

A measure of region activity is the percentage of region occupancy. This is broadly the ratio of the elapsed time a region is performing processing to the trace interval. The region occupancy time does not include those times when no messages are available, when the block loading is delayed, or when the PSB cannot be used. The last section in the Region Summary report lists all active regions for which timed events were collected and shows the calculated percentage region occupancies.

**Monitoring Application Program Elapsed Time**

The IMS Monitor can record measurements of elapsed times for each transaction and scheduling of an application program. It does this during the monitored interval while other programs are executing concurrently. Elapsed times are calculated from the start of the first DL/I (or other) call to the end of that program. You can distinguish between time spent in application code and in DL/I processing. The event intervals are illustrated in [Figure 146](#).

![Figure 146. Elapsed Time Event Intervals](#)

Within the elapsed time for a DL/I call, the wait time to obtain segment data is recorded separately. Similarly, within the elapsed time for an external subsystem call, the processing time in the external subsystem is recorded separately as the wait time. The application processing (A/P) time includes many kinds of subsidiary service beyond the machine cycles expended by the program object code—such as subroutine loading, I/O to MVS data sets, and any overlay processing. If the program is waiting to be dispatched or requires paging before it can use the real storage, these delays are also accounted for in application program processing. Because a program can execute many transactions for each schedule, the elapsed time from schedule to first call is recorded separately. This time covers the initialization performed by the application program and also includes the time for loading the program.

The elapsed times are given in the Program Summary report. [Figure 147 on page 418](#) is an example of the report. Programs are identified by their PSB name on individual lines in the report. Each line gives a summary of the activity for that PSB.
Monitoring Application Program Elapsed Time

during the measured interval. The total number of schedules, DL/I calls, transactions completed (dequeued), and waits for DL/I call I/O calls, the and external subsystem processing are given. The report line gives calculated average times for:

- Elapsed time per schedule
- Processor time per schedule
- Schedule to first DL/I call per schedule
- Elapsed time per transaction

Frequencies for calls per transaction, I/O waits per DL/I call, waits per external subsystem call, and transactions dequeued per schedule are also given. A TOTALS line summarizes all activity for the PSBs active during the monitored interval. (The PSB Dummy line reconciles any incomplete scheduling caused by a region stopping during scheduling or for a program that experiences a pseudo abend.)

**Figure 147. Program Summary Report**

You can use the Call Summary report to examine the detail of the call processing for each program, itemized by type or call and summarized for the monitor interval. An extract from the multipage output is given in Figure 148 on page 419. The calls using an I/O PCB are given first and subtotaled. Then, the total calls, of each type, against each database PCB and each external subsystem are listed. The PSB Total line marks the end of data for each program.
Monitoring I/O for Application Program DL/I Calls

The IMS Monitor report shows the total number of I/O occurrences and the total time the occurrences took for each application program executed during a monitored interval. The Program I/O report gives these two totals for all PSBs active during the monitored interval and includes the detailed breakdown of the I/O wait time as it was incurred by each PCB used by the program. Figure 149 on page 421 shows an example of the report.

The detail of the report reveals much of the contention experienced during application program processing. Each type of conflict and the number of times it occurred are recorded for each I/O PCB. The report shows the total wait time, the highest wait experienced, and the average time. Subtotals are given for each PCB under a PSB, and for all PCBs under each PSB.

The DDN/FUNC column lists the data set by DDname. The MODULE column uses a code to indicate the source of the contention. The types of conflicts and codes are shown as follows:

- **Message handling**
  - **Code**  
  - **Conflict**
    - **MFS** MFS format library directory
    - **PMM** Message format buffer pool space or control block I/O
    - **QMG** Message queue management

- **Scheduling**
  - **Code**  
  - **Conflict**
    - **BLR** Load/read from ACBLIB
    - **MSC** MPP region initialization
    - **SMN** Virtual storage management
For external subsystem calls, the elapsed time to complete the processing is considered wait time. The DDN/FUNC column indicates the external subsystem call function, as follows:

- **External subsystems**

<table>
<thead>
<tr>
<th>Code</th>
<th>Subsystem call function</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB0</td>
<td>ABORT</td>
</tr>
<tr>
<td>CT0</td>
<td>Create thread</td>
</tr>
<tr>
<td>D50</td>
<td>Terminate identify or thread, signoff</td>
</tr>
<tr>
<td>D80</td>
<td>INIT</td>
</tr>
<tr>
<td>I30</td>
<td>Identify, command, echo, terminate</td>
</tr>
<tr>
<td>I30</td>
<td>Identify, terminate subsystem</td>
</tr>
<tr>
<td>I50</td>
<td>INIT</td>
</tr>
<tr>
<td>I60</td>
<td>Resolve-in-doubt</td>
</tr>
<tr>
<td>PR0</td>
<td>Subsystem-not-operational</td>
</tr>
<tr>
<td>P10</td>
<td>Commit prepare (Phase 1)</td>
</tr>
<tr>
<td>P20</td>
<td>Commit continue (Phase 2)</td>
</tr>
<tr>
<td>SO0</td>
<td>Signon</td>
</tr>
<tr>
<td>SI0</td>
<td>Identify</td>
</tr>
</tbody>
</table>
### Program I/O Report (Part 1 of 3)

#### Figure 149.

The table below shows a detailed breakdown of I/O wait times for various programs and PCBs, including:
- **Programs:** PROGHR1A, PROGDE1A, PROGHR2B, PROGHR2A
- **PCBs:** I/O, PCB, PSB
- **Wait Times:** Total, Mean, Maximum
- **Modules:** DBH, QMG, QBLKS, VBH

<table>
<thead>
<tr>
<th>PSBNAME</th>
<th>PCB NAME</th>
<th>IWAIT</th>
<th>TOTAL</th>
<th>MEAN</th>
<th>MAXIMUM</th>
<th>DON/FUNC</th>
<th>MODULE</th>
</tr>
</thead>
</table>
| PROGHR1A I/O PCB | 122 | 2341116 | 19189 | 7079 | HOTELDBA DBH
| | 34 | 2417796 | 711115 | 3950160 | **W** | **F** | **I** |
| | 40 | 23652665 | 591316 | 2688917 | **W** | **F** | **I** |
| | 5 | 67613 | 19522 | 21124 | SHMSG QMG
| | 4 | 1108363 | 27590 | 60448 | QBLKS QMG

**PCB Total:** 131 | 2519092 | 19229

**PSB Total:** 305 | 6725063 | 20049

| PROGDE1A TRMNALDA | 20 | 624677 | 31233 | 275811 | 275811 | PI TRMNALDA.... |
| | 1 | 275811 | 275811 | 275811 | 275811 | PI TRMNALDA.... |

**PCB Total:** 21 | 900488 | 42880

| I/O PCB | 16 | 488812 | 30550 | 79980 | TRMNALDA VH |
| | 1 | 16118 | 16118 | 16118 | SHMSG QMG

**PCB Total:** 17 | 504930 | 29701

| TABLEDBA | 16 | 290471 | 18154 | 33254 | TABLEDA DBH |

**PCB Total:** 16 | 290471 | 18154

| PSB TOTAL | 54 | 1695889 | 31405 |

| PROGHR2B HOTELDBB | 8 | 698384 | 87298 | 184475 | HOTELDBB DBH |
| | 4 | 5820660 | 1455162 | 1455287 | PI HOTELDBB.... |
| | 4 | 4481024 | 1120256 | 1090075 | PI HOTELDBB.... |
| | 2 | 260817 | 130408 | 232750 | HOSINDOB VBH |
| | 7 | 106623 | 15231 | 16410 | HOSINDX8 VBH |
| | 1 | 15366 | 15366 | 15366 | HOTELDBD DBH

**PCB Total:** 26 | 11382864 | 437802

| PSB TOTAL | 26 | 11382864 | 437802

| PROGHR2A HOTELDBA | 17 | 655801 | 38576 | 366108 | HOSINDXA VBH |
| | 73 | 1836721 | 25160 | 82141 | HOTELDBA DBH |
| | 2 | 54663 | 27331 | 41975 | HOTELDB D BH |
| | 1 | 9887 | 9887 | 9887 | HOTELDBC DBH |
| | 2 | 851042 | 865635 | 856365 | HOSINDDO VBH |

**PCB Total:** 95 | 3408114 | 35874

| I/O PCB | 20 | 575847 | 28792 | 74227 | HOTELDBA DBH |
| | 21 | 370390 | 17637 | 43153 | HOSINDXA VBH |

---

Chapter 19. Interpreting IMS Monitor Reports for DCCTL 421
<table>
<thead>
<tr>
<th>PSNAME</th>
<th>PCB NAME</th>
<th>IWAITS</th>
<th>TOTAL</th>
<th>MEAN</th>
<th>MAXIMUM</th>
<th>DDN/FUNC</th>
<th>MODULE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROGHR2A 1/O PCB</td>
<td>5</td>
<td>2654544</td>
<td>390908</td>
<td>202043</td>
<td>**W F 1</td>
<td>**W F 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>32796604</td>
<td>4099575</td>
<td>9328891</td>
<td>**W F 1</td>
<td>**W F 1</td>
<td></td>
</tr>
<tr>
<td>PCB TOTAL</td>
<td></td>
<td>946237</td>
<td>23078</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSB TOTAL</td>
<td></td>
<td>136</td>
<td>4354351</td>
<td>32017</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROGPS2A LOGIMA</td>
<td></td>
<td>89</td>
<td>2046670</td>
<td>22996</td>
<td>73593</td>
<td>IMMSTR3A VBH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>612</td>
<td>53886417</td>
<td>88049</td>
<td>185674</td>
<td>IMMSTR1A VBH</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>44906</td>
<td>14968</td>
<td>20780</td>
<td>IMINDEXA VBH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCB TOTAL</td>
<td></td>
<td>704</td>
<td>55977993</td>
<td>79514</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>469</td>
<td>11742900</td>
<td>25038</td>
<td>170337</td>
<td>COMPSDA DBH</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>329</td>
<td>8198418</td>
<td>24919</td>
<td>91422</td>
<td>CPINDEXA VBH</td>
<td></td>
</tr>
<tr>
<td>PCB TOTAL</td>
<td></td>
<td>798</td>
<td>19941318</td>
<td>24989</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>47511</td>
<td>15837</td>
<td>20806</td>
<td>SHMSG QMG</td>
<td></td>
</tr>
<tr>
<td>PCB TOTAL</td>
<td></td>
<td>3</td>
<td>47511</td>
<td>15837</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSB TOTAL</td>
<td></td>
<td>1505</td>
<td>75966822</td>
<td>50476</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROGSC6C 1/O PCB</td>
<td></td>
<td>52</td>
<td>2698602</td>
<td>51896</td>
<td>473763</td>
<td>INVENTRC VBH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>70921</td>
<td>17730</td>
<td>34241</td>
<td>SHMSG QMG</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>50699</td>
<td>16899</td>
<td>24724</td>
<td>QBLKS QMG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCB TOTAL</td>
<td></td>
<td>59</td>
<td>2820222</td>
<td>47800</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>55</td>
<td>2666884</td>
<td>48488</td>
<td>210752</td>
<td>INVENTRC VBH</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>50</td>
<td>797587</td>
<td>15951</td>
<td>41706</td>
<td>ININDEXC VBH</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>119253</td>
<td>119253</td>
<td>119253</td>
<td>PI INVENTR...1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>8634</td>
<td>8634</td>
<td>8634</td>
<td>INVENTRB VBH</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>83947</td>
<td>41973</td>
<td>53936</td>
<td>INVENTRA VBH</td>
<td></td>
</tr>
<tr>
<td>PCB TOTAL</td>
<td></td>
<td>109</td>
<td>3676305</td>
<td>33727</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSB TOTAL</td>
<td></td>
<td>168</td>
<td>6496527</td>
<td>38669</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROGHR2D 1/O PCB</td>
<td></td>
<td>21</td>
<td>2285296</td>
<td>108823</td>
<td>199223</td>
<td>HOTELDBD DBH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>28</td>
<td>762370</td>
<td>27227</td>
<td>111860</td>
<td>HOTINDEXD VBH</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>11685</td>
<td>11685</td>
<td>11685</td>
<td>SHMSG QMG</td>
<td></td>
</tr>
<tr>
<td>PCB TOTAL</td>
<td></td>
<td>50</td>
<td>3059351</td>
<td>61187</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HOTELDBD</td>
<td>96</td>
<td>6279107</td>
<td>65407</td>
<td>139032</td>
<td>HOTELDBD DBH</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 149. Program I/O Report (Part 2 of 3)
The I/O waits for the calls to the I/O PCB are grouped as the first entries for a PSB. For DC DL/I calls, the data set for which the I/O took place is indicated under the DDN/FUNC heading, and the module code tells you what type of conflict caused the wait. For external subsystem calls, the function is indicated under the DDN/FUNC heading, and the module code indicates the source of the call entry. Names other than LGMSG and SHMSG can appear under the DDN/FUNC column for I/O PCBs.

If the program is designated as wait for input and has to wait for the input of the next message, the wait entry is marked **WFI under the DDN/FUNC heading and no entry appears in the MODULE column. The time spent waiting for the next input message is shown under wait time. **WFI entries are shown for information only and their values are not used to compute statistics.

**Monitoring MFS Activity**

You can obtain a summary of all activity that occurs for management of message format buffer pool use from the Message Format Buffer Pool report. The report is illustrated in Figure 150 on page 424. The data shows the counts at the start and end of the trace interval and their difference.

When message formatting occurs, the appropriate message blocks must reside in the message format buffer pool, a DIF/MID pair for input or a DOF/MOD pair for output. If the blocks are not already in the buffer, I/O to the active IMSFORMATA/B library must occur. Block retrieval can involve a prior directory lookup, or be direct, using an index kept in the pool.

Many of the counts reveal details of internal event management. The number of times there is no directory entry for a block implies extra directory lookup I/O. Delays caused by unavailable FRE entries are recorded as request-ignored counts.
Monitoring Message Queue Handling

A key resource that directly affects the efficiency of transaction processing is the message queue pool and the management of the I/O to the message queues. You can examine the activity by looking at the Message Queue Pool report. Figure 151 illustrates the report contents. Counts of activities are given at start and end of the trace interval and as the differences between start and end numbers.

Figure 150. Message Format Buffer Pool Report

Figure 151. Message Queue Pool Report
Detecting Checkpoint Effects

When a checkpoint command specifies SNAPQ, the current status of all message queues is written to the system log. This prevents any message handling on behalf of queue management. General Iwait Time Events report records the wait time incurred by the SNAPQ. Figure 152 shows the activity on the summary line QMGR SNAPQ CHECK. The number of occurrences is given with the total, average, and maximum wait times.

![Figure 152. General Reports for SNAPQ Effects](image)

Transaction Queuing Report

In addition to monitoring the efficiency of message handling, you can monitor the service provided for each application, by looking at the size of the transaction queues at each scheduling of their processing programs.

The Transaction Queuing report shown in Figure 153 on page 426 records, for each transaction, the minimum, average, and maximum counts at scheduling time. The total number of dequeued transactions (or transactions that have been fully processed) during the monitored interval is given for each transaction code. The average number of transactions processed for each scheduling is given in the DEQUEUED MEAN column.
Monitoring Line Activity

You can obtain a summary of all occurrences of activity for each BTAM line or VTAM node that handles message traffic during the monitored interval. The elapsed times and NOT-WAIT times are given in categories of total, mean, and maximum times for each communication line in the Communication Summary report. Figure 154 on page 427 illustrates this report.

Requirement: You must match which physical devices are using the line to the Stage 1 output from system definition. The line numbers are assigned sequentially, according to their physical occurrence in the Stage 1 input deck.

If your online system specifies the prefetch option for MFS blocks in the control region JCL, the last line of the report contains the statistics for all prefetch events.

You can also investigate the amount of data transmitted across BTAM lines or for VTAM nodes with the Line Functions report. Figure 155 on page 427 illustrates this report. The report distinguishes between input data and output data. The number of blocks of data and the average and maximum size of the blocks are recorded for data received by IMS and for transmitted data.
This report also includes a measure of how inactive the lines are. An inactive interval is assumed to be the difference between the time that marks the end of the last input block received and the starting time for output transmission. These occurrences of inactivity are termed turnaround intervals, and the report cumulates the number of occurrences as well as the average and maximum times associated with these intervals.

If the line is being used by an MFS supported terminal, a count of the number of requests for next page for a multipage message is recorded.

If link traffic for coupled multiple systems is recorded, a set of three reports follows the Line Functions report. These are described in the section "Interpreting IMS Monitor MSC Reports" on page 434.

Figure 154. Communication Summary Report

Figure 155. Line-Functions Report

Monitoring Message Handling Efficiency

The IMS Monitor produces both summary and detailed information on asynchronous processing in the IMS control region. The arrival of data transmitted from BTAM terminals or from VTAM triggers the processing. Application program responses also result in processing. The space in four major buffer pools and access to format, SPA, and message queue data sets are managed for the total communications traffic. Wait times are recorded when contention for pool space or I/O interrupts the processing of any of the communication tasks triggered by line activity. This information is contained in the Communication Wait report. Figure 156 on page 428 illustrates this report.

This report is complementary to the Communication Summary report in that the line number is used as an identification for the series of communication processing tasks.
IMS Internal Resource Usage

There are several summary reports that you can use to examine the level of internal contention for resources. The following sections give a brief explanation of these reports.

Pool Space Contention

The Pool Space Failure Summary report gives the number of times in each region a given amount of storage was unavailable. It shows the number of bytes and the identification of the pool as well as the number of occurrences of this failure to obtain storage. You can use this summary to determine whether you need to increase the buffer pool allocation by a system definition change or by overriding the number of buffers in the EXEC statements in the JCL.

The format of the report is shown in Figure 157.

<table>
<thead>
<tr>
<th>POOL ID</th>
<th>BYTES REQ.</th>
<th>OCCURRENCES</th>
<th>TOTAL</th>
<th>DIST.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLMP</td>
<td>8888</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Figure 157. Pool Space Failure Report

IMS Latch Conflict

The basic serialization of the task processing in IMS is controlled by ownership of an IMS latch. When different programs are executing, they compete for the ownership. If they wait for the resource, the one possessing the latch has to post the other ITASK waiting for it. You can judge the level of contention for a resource and then investigate a set of changes to relieve the pressure.

The different types of latches and the counters that exhibit the level of contention are given in the Latch Conflict Statistics report. Figure 158 on page 429 is an example of this report. The entries are organized according to the latch names.

For the latch names and abbreviations for the different types of resources being serialized, see "IMS Latch Conflict" on page 373.

When a system checkpoint is taken during the time the monitor is active, latch conflict statistics are reset to zero, thus corrupting the values presented in this report. If this situation exists, the following message will be inserted at the top of the report:

**** A CHECKPOINT OCCURRED DURING MONITOR RUN ****
**** LATCH CONFLICT STATISTICS ARE INVALID ****
**** SEE UTILITIES REFERENCE MANUAL ****

Figure 156. Communication Wait Report
However, if the master terminal operator issues the /CHECKPOINT command with the STATISTICS keyword parameter, latch conflict statistics are reset to zero, but the IMS monitor is not notified. Therefore, DFSUTR20 cannot detect that the statistics have been corrupted and does not issue this message.

**Recommendation:** Do not issue statistics checkpoints while the Monitor is running.

The counters are primarily concerned with storage management and logging services. The statistics recorded are the number of times contentions occur, that is, the resource waits for a latch.

![Latch Conflict Statistics Report](image)

**Using Frequency Distributions from IMS Monitor Output**

The reports that are derived from the IMS Monitor data records contain many summary lines where the mean time is given. If you are interested in the distribution of those timed events, rather than just average and maximum times, you can request the Report Print utility to individually record the events in a frequency distribution across a range of intervals. Some distributions are not time dependent, such as those for transaction queue loads or transmitted block sizes.

**How to Get a Frequency Distribution Output**

To request the IMS Monitor Report Print utility to gather distribution data, include a DIS input control statement. This causes all report items with an entry under a column headed MEAN to have a corresponding frequency distribution as part of the Distribution Appendix report. Each report line includes an identifying reference number under the column headed Distribution Number. You can use the reference number to locate the distribution data flagged by that number in the appendix.

The following tables show the major IMS Monitor reports and the type of frequency distributions generated for each report. Each type results in several distributions, depending on how many entries are in each section of the report. For each type of frequency distribution the data is cumulated in suitable intervals or ranges. The set of ranges used for each type is given an identifier, shown in the ID column.

Table 39 on page 430 shows the report distributions sorted by Region Summary.
Using Frequency Distributions

Table 39. Report Distributions by Region Summary

<table>
<thead>
<tr>
<th>Report Name</th>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduling and Termination</td>
<td>D1</td>
<td>Elapsed time</td>
</tr>
<tr>
<td></td>
<td>D2</td>
<td>Not wait time</td>
</tr>
<tr>
<td>Schedule end to 1st DL/I call</td>
<td>D3</td>
<td>N/A</td>
</tr>
<tr>
<td>Elapsed execution time DL/I calls</td>
<td>D4</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>D5</td>
<td>Elapsed time</td>
</tr>
<tr>
<td>External Subsystem calls</td>
<td>D6</td>
<td>Not wait time</td>
</tr>
<tr>
<td>Waits per DL/I call</td>
<td>D43</td>
<td>Elapsed time</td>
</tr>
<tr>
<td>Idle for intent Checkpoint</td>
<td>D7</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>D8</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>D20</td>
<td>Elapsed time</td>
</tr>
<tr>
<td></td>
<td>D21</td>
<td>Not wait time</td>
</tr>
</tbody>
</table>

Table 40 shows the report distributions Programs Region.

Table 40. Report Distributions by Program Region

<table>
<thead>
<tr>
<th>Report Name</th>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elapsed execution time</td>
<td>D30</td>
<td>N/A</td>
</tr>
<tr>
<td>Schedule and to 1st DL/I call</td>
<td>D31</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 41 shows the report distributions sorted by Program Summary.

Table 41. Report Distributions by Program Summary

<table>
<thead>
<tr>
<th>Report Name</th>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor time per schedule</td>
<td>D15</td>
<td>N/A</td>
</tr>
<tr>
<td>Transactions dequeued per schedule</td>
<td>D14</td>
<td>N/A</td>
</tr>
<tr>
<td>Elapsed time per schedule</td>
<td>D9</td>
<td>N/A</td>
</tr>
<tr>
<td>Schedule end to 1st DL/I call</td>
<td>D10</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 42 shows the report distributions sorted by Communication Summary.

Table 42. Report Distributions by Communication Summary

<table>
<thead>
<tr>
<th>Report Name</th>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line elapsed time</td>
<td>D18</td>
<td>N/A</td>
</tr>
<tr>
<td>Line not wait time</td>
<td>D19</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 43 shows the report distributions sorted by Line Functions.

Table 43. Report Distributions by Line Functions

<table>
<thead>
<tr>
<th>Report Name</th>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Received block length</td>
<td>D36</td>
<td>N/A</td>
</tr>
<tr>
<td>Transmitted block length</td>
<td>D37</td>
<td>N/A</td>
</tr>
<tr>
<td>Inactive intervals</td>
<td>D38</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Table 44 shows the report distributions sorted by MSC Queuing Summary.

<table>
<thead>
<tr>
<th>Report Name</th>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time in queue</td>
<td>D39</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 45 shows the report distributions sorted by Transaction Queuing.

<table>
<thead>
<tr>
<th>Report Name</th>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transactions on queue at schedule</td>
<td>D17</td>
<td>N/A</td>
</tr>
<tr>
<td>Transactions dequeued per schedule</td>
<td>D16</td>
<td>N/A</td>
</tr>
<tr>
<td>Prefetch format blocks</td>
<td>D28</td>
<td>Elapsed time</td>
</tr>
<tr>
<td></td>
<td>D29</td>
<td>Not wait time</td>
</tr>
</tbody>
</table>

Table 46 shows the report distributions sorted by Call Summary.

<table>
<thead>
<tr>
<th>Report Name</th>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSB waits per DL/I call</td>
<td>D13</td>
<td>N/A</td>
</tr>
<tr>
<td>PSB waits per external subsystem call</td>
<td>D44</td>
<td>N/A</td>
</tr>
<tr>
<td>PSB elapsed time per call</td>
<td>D11</td>
<td>N/A</td>
</tr>
<tr>
<td>PSB not wait time per call</td>
<td>D12</td>
<td>N/A</td>
</tr>
<tr>
<td>PSB external subsystem calls</td>
<td>D45</td>
<td>Elapsed time</td>
</tr>
</tbody>
</table>

Table 47 lists some distributions derived from buffer pool statistics for wait times.

<table>
<thead>
<tr>
<th>Function</th>
<th>ID</th>
<th>Module Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage</td>
<td>D22</td>
<td>SMN</td>
</tr>
<tr>
<td>Scheduler internal</td>
<td>D25</td>
<td>MSC</td>
</tr>
<tr>
<td>Queue manager I/O</td>
<td>D26</td>
<td>QMG</td>
</tr>
<tr>
<td>Block loader I/O</td>
<td>D27</td>
<td>BLR</td>
</tr>
<tr>
<td>MFS block I/O</td>
<td>D32</td>
<td>MFS</td>
</tr>
<tr>
<td>MFS directory I/O</td>
<td>D33</td>
<td>MFS</td>
</tr>
<tr>
<td>Format buffer pool space</td>
<td>D35</td>
<td>PMM</td>
</tr>
<tr>
<td>QMGR SNAPQ check</td>
<td>D42</td>
<td>None</td>
</tr>
</tbody>
</table>

How Frequency Distribution Ranges Are Defined

A set of ten intervals is defined for each summary line and the occurrences falling in each interval are cumulated. The interval ranges are preset with default end points. For example, the end points, for DL/I call elapsed time are: 0, 1000, 2000, 4000, 8000, 16000, 32000, 64000, 128000, 256000, INF (all times are in milliseconds). The default end points are chosen so that they are suitable to the event. The lower limit of the first interval always defaults to zero, and the upper limit of the tenth interval is infinity (INF).
Using Frequency Distributions

Although several types of distribution can use the same set of end points, each type is assigned a distribution identifier. You can use this to redefine the end points. To override the default end points you include an input control statement to the Report Print utility. The statement specifies the type of distribution identifier and gives the desired end point values.

**Example:** The DL/I call elapsed time end points could be respecified by:

\[
D5 \quad 0, 500, 1000, 1500, 2000, 4000, \ldots, 100000, 500000
\]

The values of the unspecified end points remains at their default values of 32000 and 64000 as does the last (INF).

Figure 159 shows a sample page from the Distribution Appendix report, which gives an example of how ranges vary with the type of distribution. The lines are arranged in pairs, with the second one recording the cumulated counts.

```
0

| IMS MONITOR | DISTRIBUTION APPENDIX | TRACE START 1993 | TRACE STOP 1993 | PAGE 020
|-------------|-----------------------|------------------|----------------|-------|
# 1........0........200000........400000........600000........800000........1000000........1200000........1400000........1600000........1800000........INF
0 0 0 0 0 0 0 0 0 0 1
# 2........0........200000........400000........600000........800000........1000000........1200000........1400000........1600000........1800000........INF
0 0 0 0 0 0 0 0 0 0 0
# 3........0........200000........400000........600000........800000........1000000........1200000........1400000........1600000........1800000........INF
0 0 0 0 0 0 0 0 0 0 0
# 4........0........200000........400000........600000........800000........1000000........1200000........1400000........1600000........1800000........INF
0 0 0 0 0 0 0 0 0 0 0
# 5........0........200000........400000........600000........800000........1000000........1200000........1400000........1600000........1800000........INF
0 0 0 0 0 0 0 0 0 0 0
# 6........0........200000........400000........600000........800000........1000000........1200000........1400000........1600000........1800000........INF
0 0 0 0 0 0 0 0 0 0 0
# 7........0........200000........400000........600000........800000........1000000........1200000........1400000........1600000........1800000........INF
0 0 0 0 0 0 0 0 0 0 0
# 8........0........200000........400000........600000........800000........1000000........1200000........1400000........1600000........1800000........INF
0 0 0 0 0 0 0 0 0 0 0
# 9........0........200000........400000........600000........800000........1000000........1200000........1400000........1600000........1800000........INF
0 0 0 0 0 0 0 0 0 0 0
# 10........0........200000........400000........600000........800000........1000000........1200000........1400000........1600000........1800000........INF
0 0 0 0 0 0 0 0 0 0 0
# 11........0........200000........400000........600000........800000........1000000........1200000........1400000........1600000........1800000........INF
0 0 0 0 0 0 0 0 0 0 0
# 12........0........200000........400000........600000........800000........1000000........1200000........1400000........1600000........1800000........INF
0 0 0 0 0 0 0 0 0 0 0
# 13........0........200000........400000........600000........800000........1000000........1200000........1400000........1600000........1800000........INF
0 0 0 0 0 0 0 0 0 0 0
# 14........0........200000........400000........600000........800000........1000000........1200000........1400000........1600000........1800000........INF
0 0 0 0 0 0 0 0 0 0 0
# 15........0........200000........400000........600000........800000........1000000........1200000........1400000........1600000........1800000........INF
0 0 0 0 0 0 0 0 0 0 0
# 16........0........200000........400000........600000........800000........1000000........1200000........1400000........1600000........1800000........INF
0 0 0 0 0 0 0 0 0 0 1
```

**Figure 159. Distribution Appendix Report**

**Default Values of Distribution Definitions**

Using an identifier provided in the frequency distribution tables (Table 39 on page 430 through Table 46 on page 431) and the Wait Time Distributions table (Table 47 on page 431) you can determine the default end points for the distribution by locating it in the following list:

- **D1, D2, D5, D6, D9, D10, D11, D12, D15, D18, D19, D20, D21, D22, D25, D27, D28, D29, D30, D31, D43, and D45**
  - 0, 1000, 2000, 4000, 8000, 16000, 32000, 64000, 128000, 256000, INF
- **D3**
  - 0, 50000, 100000, 150000, 200000, 250000, 300000, 350000, 400000, 450000, INF
- **D4**
  - 0, 200000, 400000, 600000, 800000, 1000000, 1200000, 1400000, 1600000, 1800000, INF
Interpreting Distribution Appendix Output

You can use the detailed output in the Distribution Appendix when you suspect an unusual combination of events was reported in a report summary line. Usually, the average and maximum times or counts are sufficient to highlight a resource usage problem. However, if you suspect the mean value to be masking an unusual distribution you can draw on the detail contained in the IMS Monitor output records.

For example, suppose you are investigating a change in the scheduling algorithm for a particular transaction and need to know how many transactions were able to be processed for each scheduling of an application program. Figure 160 shows the processed transactions in a histogram:

![Histogram of Transactions Processed](image)

**Figure 160. Number of Transactions Processed For Each Scheduling Of An Application Program**

The average is 2.5 transactions per schedule. The distribution in Figure 160 suggests that many schedules were able to process only one or two transactions, and few schedules significantly exhausted the queue. The distribution data for the histogram is as follows:
Using Frequency Distributions

<table>
<thead>
<tr>
<th>Number of schedules</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6-10</th>
<th>&gt;10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transactions dequeued</td>
<td>8</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

The Distribution Appendix presents the histogram data in the form of two lines:
- The first line shows the intervals, prefixed by a cross reference to an individual line on the earlier output.
- The second line gives the number of events occurring in those intervals.

This data appears as follows:

```
  # 950B...0...1...2...3...4...5...10...15...30...90...INF
  8 7 0 0 2 1 0 0 0 0 0
```

The cross reference 950B points to a unique report line. For example, the Transaction Queuing report on the appropriate line for the transaction of interest shows 950A,B under the column headed DISTRIBUTION NUMBER. Use the reference number 950B to locate the data in the Distribution Appendix. The 950A reference points to the data for the number of transactions in the queue at schedule time.

Interpreting IMS Monitor MSC Reports

The IMS Monitor Report Print program includes three reports that highlight message events caused by system coupling. They are:
- MSC Traffic report
  - For the monitor interval, this report shows the enqueue and dequeue counts of messages that use the various link paths.
- MSC Summaries
  - This report shows summaries of:
    - The traffic queues for each input transaction name
    - The traffic queues for each destination name
    - The traffic queues for each link number
    - The traffic queues for each destination system
- MSC Queuing Summary report
  - This report is generated only when intersystem messages are queued on the local system before being sent to the destination system. The local system must be an intermediate system. This report shows:
    - Maximum time messages spend in queues
    - Average time messages spend in queues
    - Maximum queue lengths
    - Maximum queue counts
    - Total number of messages queued for all links in which the local system participates

All three of the reports can have entries in the Distribution Appendix. You can examine the frequency distributions of the traffic if you suspect unusual transmission patterns.

Determining Cross System Queuing

The MSC Traffic report reveals the individual queue loads for all traffic between partner systems of which the monitored system is the local system. The report lists all the unique system identification numbers (SIDs) that are defined for
Interpreting IMS Monitor MSC Reports

communications for that local system. It then summarizes the total messages queued and dequeued for each combination of the following variables:

- Input name (terminal or program that was a message source)
- Destination name (terminal or program)
- Input system (SID)
- Destination system (SID)
- Link number
- Link type (BSYN, MTM, CTC, or VTAM)

Figure 161 illustrates this report. If a message originates in the local system, its presence is accounted for in the dequeue counts only. Messages with local destinations appear only in the enqueue count.

<table>
<thead>
<tr>
<th>INPUT NAME</th>
<th>LOCAL SID</th>
<th>LINK TYPE</th>
<th>ENQUEUE COUNT</th>
<th>DEQUEUE COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSWT3685</td>
<td>2</td>
<td>C-C</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DSWT6161</td>
<td>3</td>
<td>C-C</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>SC62</td>
<td>3</td>
<td>C-C</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>DSWT3838</td>
<td>3</td>
<td>C-C</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>SC6Y</td>
<td>2</td>
<td>C-C</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>DSWT4618</td>
<td>3</td>
<td>C-C</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>SC62</td>
<td>3</td>
<td>C-C</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>DSWT3903</td>
<td>3</td>
<td>C-C</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>SC2Z</td>
<td>3</td>
<td>C-C</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>DSWT5418</td>
<td>3</td>
<td>C-C</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>SC2Z</td>
<td>3</td>
<td>C-C</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>DSWT4673</td>
<td>3</td>
<td>C-C</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>SC2Z</td>
<td>3</td>
<td>C-C</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>DSWT5141</td>
<td>45</td>
<td>45</td>
<td>VTAM</td>
<td>0</td>
</tr>
<tr>
<td>PS3X</td>
<td>45</td>
<td>145</td>
<td>45 VTAM</td>
<td>1</td>
</tr>
<tr>
<td>DSWT4391</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>SC4Y</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>DSWT3324</td>
<td>17</td>
<td>17</td>
<td>VTAM</td>
<td>1</td>
</tr>
<tr>
<td>ITY</td>
<td>17</td>
<td>117</td>
<td>17 VTAM</td>
<td>1</td>
</tr>
<tr>
<td>DSWT4781</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>SC2Z</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>DSWT3525</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>SC6Z</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>DSWT4542</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>SC6Y</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>DSWT4796</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>SC2Z</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>DSWT4782</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>SC62</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>DSWT4633</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>SC6Y</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>DSWT3655</td>
<td>12</td>
<td>12</td>
<td>VTAM</td>
<td>0</td>
</tr>
<tr>
<td>SC6U</td>
<td>12</td>
<td>112</td>
<td>12 VTAM</td>
<td>1</td>
</tr>
<tr>
<td>DSWT3892</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>SC6Y</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>DSWT3338</td>
<td>4</td>
<td>4</td>
<td>VTAM</td>
<td>0</td>
</tr>
<tr>
<td>SC2U</td>
<td>4</td>
<td>104</td>
<td>4 VTAM</td>
<td>1</td>
</tr>
<tr>
<td>DSWT4681</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>SC4Z</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>DSWT4482</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>SC6Y</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>DSWT4902</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>SC2Z</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>DSWT4558</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>SC6Z</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>DSWT3925</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**TOTAL TRAFFIC**

1353 1359

Figure 161. MSC Traffic Report
Assessing the Effect of Link Loading

The MSC Summaries report shows you the enqueue and dequeue activity for messages that are handled by the local system but are part of the multiple system coupling traffic. The report format is illustrated in Figure 162.

The first set of queuing counts shows how many transactions of each type were entered in the monitor interval, and how many were subsequently dequeued.

The second set of counts summarizes the total traffic for each destination name. You can distinguish the primary transactions and responses by the resource names and examine the relative servicing of the link transmissions using the difference between the enqueue and dequeue counts.

The third set of counts lists the active links by link number, and you can determine if there is a buildup on the link by the difference in the enqueue and dequeue counts.

The fourth set of counts records the traffic that is going to other systems by whatever link path. You can judge by the difference in enqueue and dequeue counts whether the overall pattern of link priorities to one or more systems is causing buildup of cross system traffic.
Assessing Link Queuing Times

The MSC Queuing Summary report provides information about intersystem message traffic only. You can use the sample of traffic recorded in the IMS Monitor interval to examine the maximum and average time messages spend in queues waiting to be sent on active links. You can detect whether the link priorities are causing undue delay of primary messages through the intermediate system or whether there is a buildup of responses. The report shows the logical link paths for this system which is an intermediate system. Each incoming link number shows the number of messages that are queued before transmission on their specified outward bound link number. The maximum queue count is given as well as the maximum and average time on the intermediate system queues.

The report is illustrated in Figure 163.

<table>
<thead>
<tr>
<th>LINK NO.TYPE</th>
<th>LINK NO.TYPE</th>
<th>MESSAGES</th>
<th>MAX.Q LENGTH</th>
<th>MAX.Q TIME</th>
<th>MEAN Q TIME</th>
<th>DIST. NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>46 VTAM</td>
<td>46 VTAM</td>
<td>12</td>
<td>1</td>
<td>31468</td>
<td>9521</td>
<td>1475</td>
</tr>
<tr>
<td>49 VTAM</td>
<td>49 VTAM</td>
<td>15</td>
<td>1</td>
<td>30235</td>
<td>6040</td>
<td>1563</td>
</tr>
<tr>
<td>50 VTAM</td>
<td>50 VTAM</td>
<td>10</td>
<td>1</td>
<td>13042</td>
<td>5521</td>
<td>1539</td>
</tr>
<tr>
<td>48 VTAM</td>
<td>48 VTAM</td>
<td>9</td>
<td>1</td>
<td>7730</td>
<td>4429</td>
<td>1762</td>
</tr>
<tr>
<td>47 VTAM</td>
<td>47 VTAM</td>
<td>8</td>
<td>1</td>
<td>10035</td>
<td>5791</td>
<td>1998</td>
</tr>
<tr>
<td>TOTALS...</td>
<td></td>
<td>54</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 163. MSC Queuing Summary Report

Extracting Multiple System Transaction Statistics

You can use the Log Transaction Analysis utility to obtain counts of the message traffic both in local systems and between systems. The transmissions over the different types of physical links can also be examined. The activity is summarized for each step of the logical link paths. You must provide IMS system log input that reflects all partner system activity, that is, sets of system logs for each MSC system. To coordinate the sets of individual system logs you use the Log Merge utility. Up to nine separate system logs can be merged; each log must be the output of a uniquely identified IMS system with MSC installed.

Controlling the Log Merge

To control the log output, you must:

- Choose the required systems that take part in the logical link paths you are examining.
- Coordinate the series of input logs for each system so that they cover a similar time span.
- Specify a start and stop time for the Log Merge utility control statements if you are to sampling the cross system processing for a particular 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 occurs between the initial and final events present on the first log.

- Specify MSG to select log records that are suitable for the transaction analysis step. (ALL records is the default, but this means the DL/I activity for several systems is included in the utility input and this could cause extended processing time.)
Interpreting the Transaction Analysis Report

You can use the Log Analysis report produced by the Log Transaction Analysis utility to obtain the following statistics for individual transactions processed in any system:

- The total response time
- The time on input and output queues
- The processing time

Chapter 21, “Interpreting Statistical-Analysis and Log-Transaction Reports,” on page 455 defines the format of the detailed report records for this report, provides a list of processing type codes, and shows an illustration of the report. The absence of times for a message GU call or MPP termination in the report lines indicate an input source or intermediate system report line.

The processing type field is used to interpret the detailed report lines. The S code indicates that the line shows a send or receive event for the transaction. You can trace the progress of a cross system conversation using the codes C, D, P, X, and Y.

The report includes a column headed SYS ID (located after the GU column that indicates the message queue time). The number shown in a report line under the ID heading matches the sequence in which log input was fed to the Log Merge utility. The field corresponds to starting position 102, the 3-digit field named SYSTEM ID, in the detailed report records.

You can use the sort step to order the report records by system ID within transaction code, or some other convenient sequence, rather than by the default which is the overall input sequence.
Chapter 20. Interpreting //DFSSTAT Reports

The //DFSSTAT reports show you how many DB and DC calls are issued by an application program and describe buffering activity during the application’s execution. The reports are written when the application program terminates.

In this Chapter:
- “JCL Description for //DFSSTAT”
- “Report Descriptions for //DFSSTAT”

JCL Description for //DFSSTAT

To get the reports, you must put a //DFSSTAT DD statement in the JCL of your batch region or online dependent region. The following is an example of a //DFSSTAT DD statement.

```
//DFSSTAT DD SYSOUT=A
```

Recommendation: Although it is supported, do not include the //DFSSTAT DD statement in the JCL for an MPP region. If you do not include the //DFSSTAT DD statement, you avoid the overhead and large amount of output that results from creating one set of reports each time a short-running MPP terminates.

Report Descriptions for //DFSSTAT

The //DFSSTAT reports are as follows:
- PST-Accounting report
- VSAM-Buffer-Pool report (for batch regions only)
- OSAM-Buffer-Pool report (for batch regions only)
- Sequential-Buffering-Summary report
- Sequential-Buffering-Detail report

PST-Accounting Report

This report shows how many DB and DC calls are issued by an application program.

Fields in the Report

Figure 164 on page 440 is an example of a PST-Accounting report and shows the names of each field. Each field in this report represents one type of DB or DC call. For example, the DB GU CALLS field shows how many database Get Unique calls were issued by the application.
PST-Accounting Report

*** PST ACCOUNTING STATISTICS ***

DB GU CALLS 2
DB GN CALLS 2
DB GNP CALLS 1
DB GHU CALLS 1
DB GHN CALLS 1
DB GHNP CALLS 1
DB ISRT CALLS 2
DB DLET CALLS 1
DB REPL CALLS 1
DB CALLS (TOTAL) 12
DB DEQ CALLS 1
MSG GU CALLS 2
MSG GN CALLS 2
MSG CHNG CALLS 4
MSG ISRT CALLS 8
MSG PURGE CALLS 4
MSG CMD CALLS 1
MSG GCMD CALLS 1
MSG AUTH CALLS 1
MSG SETO CALLS 4
SYS APSB CALLS 0
SYS DPSB CALLS 0
SYS GMSG CALLS 2
SYS ICMD CALLS 1
SYS RCMO CALLS 2
SYS CHKP CALLS 0
SYS XRST CALLS 0
SYS ROLB CALLS 1
SYS ROLS CALLS 2
SYS SETS CALLS 1
SYS SETU CALLS 1
SYS INIT CALLS 1
SYS INQY CALLS 3
SYS LOG CALLS 1

Figure 164. PST-Accounting Report

VSAM-Buffer-Pool Report

The VSAM-Buffer-Pool report describes VSAM buffer pool activity during the execution of an application program. A separate report is written for each VSAM sub-pool. The last VSAM-Buffer-Pool report summarizes the buffering activity in all the VSAM sub-pools used by the application.

This report is written only for applications that you run in batch regions.

Fields in the Report

Figure 165 on page 441 is an example of a VSAM-Buffer-Pool report. This report is identical to the VSAM-Buffer-Pool report written by the DB monitor, with the following exceptions:

- Although the field names in the DB Monitor’s VSAM Buffer Pool report are preceded by “NUMBER OF”, the fields in both reports have the same meaning.
- The //DFSSTAT VSAM-Buffer-Pool report does not keep track of the start trace and end trace times. This is unnecessary because information is always gathered for the //DFSSTAT reports from the beginning to the ending of the application’s execution.
- The //DFSSTAT VSAM-Buffer-Pool report contains a “TOTAL I/O OPERATIONS” field, which is the sum of the following:
The number of times a CI was read into the buffer from the database
(CONTROL INTERVAL READ FROM EXTERNAL STORAGE field in the report)

The number of times a buffer was written to the database (VSAM WRITES
INITIATED BY IMS/ESA field in the report)

The number of times a buffer was written to the database so a new CI could
be read into the buffer (VSAM WRITES TO MAKE SPACE IN THE POOL field
in the report).

The //DFSSTAT VSAM-Buffer-Pool report includes a summary report. The
summary report is preceded by SUBPOOL BUFFER SIZE=ALL. It contains a
summary of read and write information for all VSAM Buffer Pool reports.

The fields that represent I/O operations are highlighted on the left by an asterisk (*).

Related Reading: See IMS Version 8: Utilities Reference: Database and
Transaction Manager for a description of the various fields in the report.

Using the Report
The primary use of the VSAM-Buffer-Pool report, an example is shown in
Figure 165 is to see how many I/O operations were issued in each VSAM sub-pool.

*** VSAM BUFFER POOL STATISTICS ***

| Field                                      | Value  
|--------------------------------------------|--------
| FIX INDEX/BLOCK/DATA                      | N/Y/N  
| SHARED RESOURCE POOL ID                   | VPL1   
| SHARED RESOURCE POOL TYPE                 | D      
| SUBPOOL BUFFER SIZE                       | 4,096  
| TOTAL BUFFERS IN SUBPOOL                  | 1,000  
| TOTAL HIPERSPACE BUFFERS IN SUBPOOL       | 50     

| Field                                      | Value  
|--------------------------------------------|--------
| RETRIEVE BY RBA CALLS                     | 370    
| RETRIEVE BY KEY CALLS                     | 187583 
| LOGICAL RECORDS INSERTED INTO ESDS         | 310    
| LOGICAL RECORDS INSERTED INTO KSDS         | 9823   
| LOGICAL RECORDS ALTERED IN THIS SUBPOOL   | 0      
| TIMES BACKGROUND WRITE FUNCTION INVOKED    | 0      
| SYNCHRONIZATION CALLS RECEIVED            | 29923  
| PERM WRT ERROR BUFFS NOW IN THE SUBPOOL   | 0      
| LARGEST NBR OF PERM ERR BUFFS EVEN IN THE SUBPL | 0   
| VSAM GET CALLS ISSUED                     | 0      
| VSAM SCHBFR CALLS ISSUED                  | 189290 
| CONTROL INTERVAL REQUESTED ALREADY IN POOL| 0      
| *CONTROL INTERVAL READ FROM EXTERNAL STORAGE| 51238 
| *VSAM WRITES INITIATED BY IMS/ESA          | 138637 
| *VSAM WRITES TO MAKE SPACE IN THE POOL     | 9288   
| VSAM READS FROM HIPERSPACE BUFFERS        | 0      
| VSAM WRITES FROM HIPERSPACE BUFFERS       | 0      
| FAILED VSAM READS FROM HIPERSPACE BUFFERS | 0      
| FAILED VSAM WRITES TO HIPERSPACE BUFFERS  | 0      

*TOTAL I/O OPERATIONS                     | 199163 

Figure 165. VSAM-Buffer-Pool Report

OSAM-Buffer-Pool Report
The OSAM-Buffer-Pool report describes the OSAM buffer pool activity during an
application’s execution. This report is only written for programs running in batch
regions.
Fields in the Report

Figure 166 on page 443 is an example of an OSAM-Buffer-Pool report.

The OSAM-Buffer-Pool report is identical to the Database-Buffer-Pool report written by the DB monitor, with the following exceptions:

• The field names in the IMS Monitor’s Database-Buffer-Pool report are preceded by “NUMBER OF”, although the fields have the same meaning in both reports.

• The //DFSSTAT OSAM-Buffer-Pool report does not keep track of the start trace and end trace times. This is unnecessary because information is always gathered for the //DFSSTAT reports from the beginning to the ending of the application’s execution.

• In addition, the //DFSSTAT OSAM-Buffer-Pool report contains a “TOTAL I/O OPERATIONS” field, which equals the sum of the following fields:
  – READ REQUESTS ISSUED
  – OSAM WRITES ISSUED
  – QUEUED WRITES ISSUED
  – FORMAT LOGICAL CYLINDER REQUESTS
  – BISAM READS OR QISAM SETLS.

These fields represent I/O operations and are highlighted on the left by an asterisk (*).

Related Reading: Refer to IMS Version 8: Utilities Reference: Database and Transaction Manager for a description of the various fields in the OSAM-Buffer-Pool Report.

Using the Report

The primary use of the OSAM-Buffer-Pool report, an example is shown in Figure 166 on page 443, is to see how many OSAM I/O operations were issued. This report does not, however, show Sequential Buffering (SB) related information.
Sequential-Buffering-Summary Report

The Sequential-Buffering-Summary report provides an overview of SB-related information for the application. (VSAM-related information is not included.)

Sequential-Buffering Summary Report Fields

Figure 167 on page 445 is an example of a Sequential-Buffering-Summary report.

The first part of this report shows why Sequential Buffering was or was not used. This part of the report tells you whether:

- A SBONLINE control card was provided in DFSVSMxx (this applies only to IMS DC environments).
- A /STOP SB command was in effect when the application program started (this applies only to IMS DC environments).
- The SB Initialization Exit Routine (DFSSBUX0) disallowed use of SB.
- The SB Initialization Exit Routine (DFSSBUX0) requested conditional activation of SB by default.
- At least one SB= keyword was provided during PSBGEN.
- The //DFSCTL file contained at least one SBPARM control statement that applied to the application program.
- SBPARM control cards have been read. If the answer is Yes, the following statistics indicate what SBPARM keywords were used. This can be helpful in determining why sequential buffering was or was not used for the application program.
- At least one PSB= keyword was specified on an SBPARM control card and it matched the PSB used by the application.
PST-Accounting Report

- At least one DB= keyword was specified on an SBPARM control card where the PSB matched or was not specified, and the database matched one used by the application.
- At least one PCB= keyword was specified on an SBPARM control card where the PSB and DB matched or were not specified, and the PCB name matched one used by the application.
- At least one DD= keyword was specified on an SBPARM control card where the PSB, DB, and PSB matched or were not specified, and the DDNAME matched one used by the application.
- Whether SBPARM control cards have been read. If the answer is “yes,” the following statistics indicate what SBPARM keywords were used. This is helpful in determining why sequential buffering was or was not used for the application program.
- At least one PSB= keyword was specified on a SBPARM control card and it matched the PSB used by the application.

The other fields in the report are as follows:

**NUMBER OF SEARCH REQUESTS ISSUED BY THE OSAM BH**
This field shows you how many times the OSAM buffer handler asked the SB buffer handler to search the SB buffer pools for a specific OSAM block.

The value in this field is equal to the number of OSAM random read I/O operations that would have been issued without SB.

**NUMBER OF READ I/O**
These fields show you the number of OSAM random and sequential read I/O operations it took to satisfy requests made by the application program. The sum of these two numbers is the total number of OSAM read I/O operations issued on behalf of the application. You can subtract this sum from the NUMBER OF SEARCH REQUESTS ISSUED BY THE OSAM BH field to calculate how many read I/O operations you saved by using SB.

**NUMBER OF BLOCKS READ**
These fields tell you how many OSAM data set blocks were read to satisfy requests from the application program. These fields show you:
- The total number of blocks read
- The number and percentage of blocks read with a random read
- The number and percentage of blocks read with a sequential read

If the percentage of blocks read with a sequential read is high, SB probably helped reduce the elapsed time of the application program.

**PERCENT READ PER SEARCH REQUEST**
This field shows you the number of read I/O operations issued by the SB buffer handler expressed as a percentage of the number of times the OSAM buffer handler asked the SB buffer handler to search for a block.

A low percentage indicates that many of the search requests were satisfied without issuing an I/O operation. Therefore, a low number in this field shows that SB probably helped reduce the elapsed time of the application program.

**NUMBER OF SEQUENTIAL I/O ERRORS**
This field tells you the number of sequential reads that resulted in I/O errors. When an I/O error is detected during a sequential read, IMS increments this field and marks the 10 SB buffers involved in the read as invalid. Then IMS issues a random read for the block that was requested by the OSAM buffer handler.
Using the Report
From an SB Summary report, an example is shown in Figure 167, you can determine if the application benefited from the use of SB. When using this report, pay particular attention to these fields:

- NBR BLOCKS READ SEQUENTIALLY and PCT OF TOTAL
- PERCENT READ PER SEARCH REQUEST

*** SEQUENTIAL BUFFERING SUMMARY FOR THE APPLICATION ***

DFSSBUXO DISALLOWED USAGE OF SB: NO
DFSSBUXO REQUESTED CONDITIONAL SB ACTIVATION: NO
AT LEAST ONE SB= KEYWORD IN PSB: YES
AT LEAST ONE SBPARM CONTROL STMT FOR APPLICATION: NO
SBPARM CONTROL CARD(S) READ FROM //DFSCTL: YES
AT LEAST ONE SBPARM PSB= SPECIFIED THAT MATCHED PSB: YES
AT LEAST ONE SBPARM DB= SPECIFIED THAT MATCHED DB: YES
AT LEAST ONE SBPARM PCB= SPECIFIED THAT MATCHED PCB: NO
AT LEAST ONE SBPARM DD= SPECIFIED THAT MATCHED DD: NO

NUMBER OF SEARCH REQUESTS ISSUED BY OSAM BH:
SEARCH 2,213

NUMBER OF READ I/O:
RANDOM READ 686
SEQUENTIAL READ 652

NUMBER OF BLOCKS READ:
TOTAL NUMBER BLOCKS READ 7,206
NBR BLOCKS READ AT RANDOM 686 PCT OF TOTAL: 9
NBR BLOCKS READ SEQUENTIALLY 6,520 PCT OF TOTAL: 9
PERCENT READ PER SEARCH REQUEST 60.46
NUMBER OF SEQUENTIAL I/O ERRORS 0

Figure 167. Sequential-Buffering-Summary Report

Sequential-Buffering-Detail Report

This report gives you detailed information about how SB was used for a particular SB buffer pool. A separate report is created for each SB buffer pool used by the application program.

Each report consists of three pages, A, B, and C. Summary information is contained on page A. More detailed information is found on pages B and C.

Each of the following sections contains an example of the relevant page (A,B or C) of the Sequential-Buffering-Detail report.

Fields on Page A
Figure 168 on page 446 shows an example of page A of the report.
You can use the first section of this page to identify the SB buffer pool, database PCB, and DD name this report applies to. There can be more than one SB buffer pool (and, thus, more than one report) created for a particular database PCB and DD name. This can happen if the PCB is involved in a logical relationship.

This section contains the following information:

- The PSB name.
- The DBD name (as coded in the PCB macro during PSBGEN).
- The PCB label (as coded in the PCB macro during PSBGEN).
- The relative number of the database PCB within the PSB.
- A unique identifier of the data set group control block within the database PCB. (You can use this number to uniquely identify the SB buffer pool when more than one SB buffer pool has been created for the same PCB and DD name.)
- The DD name.
- The type of database organization (HDAM, HIDAM, PHDAM, PHIDAM, or HISAM).
- The type of database data set. This can be one of three values:
  - *INDX indicates this report applies to a data set used as an index.
  - *PSDATA indicates this report applies to a data set containing data accessed according to its primary sequence.
  - *SSDATA indicates this report applies to a data set containing data accessed according to a secondary index sequence or by crossing a logical relationship.
- The number of buffer sets in the SB buffer pool. The default number of buffer sets is four. You might have changed this default, however, with a SBPARM control statement or in the SB Initialization Exit routine.

**NUMBER OF SEARCH REQUESTS ISSUED BY OSAM BH**

This field shows you the number of OSAM random read I/O operations that would have been issued without using SB.
NUMBER OF READ I/O
These fields show you the number of read I/O operations that were actually issued in this SB buffer pool. The fields tell you:
- The total number of OSAM read I/O operations
- The number of random reads
- The number of synchronous sequential reads
- The number of overlapped (asynchronous) sequential reads

You can use these fields to calculate the percentage of each of the three types of read I/O operations that were used for this database PCB and DD name. You can also subtract the total number of OSAM read I/O operations from the NUMBER OF SEARCH REQUESTS ISSUED BY OSAM BH field to determine the number of read I/O operations you saved by using SB.

NUMBER OF BLOCKS READ
These fields show you how many blocks were read by each type of read I/O operation. The fields tell you:
- The total number of blocks read
- The number and percentage of blocks read by random reads
- The number and percentage of blocks read by synchronous sequential reads
- The number and percentage of blocks read by overlapped (asynchronous) sequential reads

A high percentage of blocks read with sequential reads tells you that SB probably helped the application program run faster, at least while processing this database PCB and data set.

A high percentage of blocks read with random reads, however, might indicate:
- A large amount of random processing by the application program
- The OSAM data set associated with this database PCB needs reorganizing

If many blocks were read with random reads, and if the program was processing the database sequentially, you can sometimes get better buffering performance by increasing the number of buffer sets (for example, increase the value of the BUFSIZE parameter on the SBPARM control statement in //DFSCTL). After increasing the number of buffer sets, observe how the number reported in the NUMBER OF READ I/O field changes the next time the application is executed.

A small percentage of blocks read at random sometimes indicates that you can reduce the number of buffer sets to save virtual storage space.

AVERAGE I/O WAIT TIMES (MILLIS)
These fields tell you the average I/O wait times for each of the three types of read I/O operations. These times show the average time the application program waited for a read I/O operation to complete before it could process the data being read.

The times in these fields are in milliseconds. A millisecond is one thousandth of a second (in other words, 50 milliseconds equals 0.050 seconds).

These fields show you:
- The average I/O wait time for random reads
- The average I/O wait time for synchronous sequential reads
- The average I/O wait time for overlapped (asynchronous) sequential reads
These times are only measured when SB is active or IMS is monitoring the I/O reference pattern.

**Fields on Page B**

Figure 169 shows an example of page B of the report.

<table>
<thead>
<tr>
<th><strong>REFERENCES IN BUFFER-SETS:</strong></th>
<th><strong>RATIO</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>REFERENCES IN RANDOM SRAN CBS:</strong></td>
<td><strong>RATIO</strong></td>
</tr>
<tr>
<td><strong>REFERENCES IN RANDOM SRAN CBS WHICH HAVE BEEN CONVERTED:</strong></td>
<td><strong>NUMBER</strong></td>
</tr>
<tr>
<td></td>
<td><strong>PCT OF STOLEN RANDOM SRAN</strong></td>
</tr>
<tr>
<td><strong>DISTRIBUTION OF REFERENCES IN BUFFER-SETS</strong></td>
<td></td>
</tr>
<tr>
<td>REFERENCE COUNT</td>
<td>NBR OF OCCURRENCES</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>551</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>=&gt; 10</td>
<td>94</td>
</tr>
</tbody>
</table>

| **DISTRIBUTION OF REFERENCES IN RANDOM SRAN CBS** | | | |
| REFERENCE COUNT | NBR OF OCCURRENCES | PCT OF OCCURRENCES | ACCUMUL. PCT |
| 0 | 0 | .00 | .00 |
| 1 | 0 | .00 | .00 |
| 2 | 1 | 100.00 | 100.00 |

**Figure 169. Sequential-Buffering-Detail Report Page B**

This page can be used to evaluate the efficiency of the SB algorithm that monitored the I/O reference pattern for this SB buffer pool. By analyzing this page, you can answer these questions:

- How efficient were the decisions to issue sequential reads? This question can be answered by the REFERENCES IN BUFFER SETS field.
- How efficient were the decisions to issue random reads? This question can be answered by the REFERENCES IN RANDOM SRAN CBS field and the RANDOM SRAN CBS WHICH HAVE BEEN CONVERTED field.

The fields on Page B reflect buffering activity when SB was active. Buffering activity when IMS was only monitoring the I/O reference pattern is not included.

**REFERENCES IN BUFFER SETS**

This field shows you how many times blocks read with a sequential read were referenced by the OSAM buffer handler. This ratio is calculated as follows:

The number of references in buffer sets divided by the number of blocks read by sequential read.

For example, a ratio of 1.00 means that, on the average, each block read by a sequential read was referenced once. A ratio of 0.50 means that, on the average, only half the blocks read by a sequential read were referenced once.

In general, a high ratio (for example, 0.85) shows you that the decisions to issue sequential reads were efficient and probably helped reduce the execution time.
time of the application. A low ratio (for example, 0.30) shows you that the benefit of issuing sequential reads was smaller because many of the blocks read were never referenced.

REFERENCES IN RANDOM SRAN CBS
This field shows you the effectiveness of the decisions to issue random reads. To understand what this ratio means, you must first understand how the SB buffer handler uses control blocks to track the I/O reference pattern.

The SB buffer handler uses a control block called SB range (SRAN) to track the I/O reference pattern within a range of 10 consecutive blocks. Each SRAN has a reference counter that shows how many times the OSAM buffer handler requested a block contained in the set of 10 blocks tracked by the SRAN.

There are two types of SRAN control blocks. One type, called a sequential SRAN, maintains counts for 10 consecutive blocks that have been read with a sequential read. A second type, called a random SRAN, maintains counts for 10 consecutive blocks that have been referenced in a random pattern. There is one sequential SRAN for each buffer set in an SB buffer pool; the number of random SRANs is twice the number of sequential SRANs.

The SRANs are chained together on a “use chain,” that is, they are chained together in the order in which they have been most or least recently used. Each type of SRAN has its own use chain. Most recently used SRANs are at the top of the use chain; least recently used SRANs are at the bottom. Whenever a SRAN is needed to track a set of blocks that are not currently being tracked, the SB buffer handler selects a SRAN from the bottom of the use chain. This is because ranges of consecutive blocks that have not been referenced recently are less likely to be referenced again in the near future.

Each time a random SRAN is reused, the reference count maintained in the SRAN is recorded for later use. When the application program terminates, these reference counts are used to calculate the number in this field. The number is calculated as follows:

\[
\text{Ratio} = \frac{X}{(Y \times 10)}
\]

where,

\( X \) Total number of random references in random SRANs (this number is usually equal or close to the number of random reads in the sets of blocks tracked by random SRANs)

\( Y \) Total number of reused random SRANs

A low ratio (for example, 0.15) in this field means that on the average, the SB buffer handler correctly recognized random I/O reference patterns. A ratio of 0.15 means that the SB buffer handler issued an average of 1.5 random reads for blocks tracked by a random SRAN. Issuing 1.5 random reads for a set of 10 consecutive blocks is normally more efficient than issuing one sequential read for the 10 consecutive blocks.

A high ratio (for example, 0.50) in this field probably indicates that the SB buffer handler often mistook sequential reference patterns as random reference patterns. A ratio of 0.50 means that the SB buffer handler issued an average of 5 random reads for blocks tracked by a random SRAN. Issuing 5 random reads for a set of 10 consecutive blocks is normally less efficient than issuing one sequential read for the 10 consecutive blocks.
RANDOM SRAN CBS WHICH HAVE BEEN CONVERTED
These fields are another measurement of the SB buffer handler’s effectiveness in analyzing the I/O reference patterns.

Sometimes the SB buffer handler interprets a reference to a set of consecutive blocks as a random pattern, then, after some additional references, detects that the set of blocks is actually being referenced sequentially. In this case, the SB buffer handler first issues several random reads in a set of consecutive blocks and later issues a sequential read for the same set of blocks. When this occurs, the random SRAN that tracks the set of blocks is converted to a sequential SRAN. If the SB buffer handler had originally read the set of blocks with a sequential read, the cost of issuing several random reads would have been avoided.

The fields in this section are:

NUMBER
This tells you how many times random SRANs were converted to sequential SRANs during the application program’s execution.

PCT OF STOLEN RANDOM SRAN
The value in this field expresses the number of converted SRANs as a percentage of the number of times the SB buffer handler had to acquire a random SRAN from the use chain. A high percentage (for example, 40 percent) probably indicates that the SB buffer handler often mistook a sequential reference pattern as random reference pattern.

DISTRIBUTION OF REFERENCES IN BUFFER SETS
This table shows you more detailed information about the ratio reported in the REFERENCES IN BUFFER SETS field.

For example, the REFERENCES IN BUFFER SETS field might indicate that 80 percent (0.80) of the blocks in the SB buffer sets were referenced. What does this mean? It could mean that 20 percent of the time none of the blocks in a buffer set were referenced. Or instead, it could mean that 100 percent of the time only 8 out of 10 blocks in the buffer sets were referenced. You can answer this question by analyzing this table.

The fields in this table are:

REFERENCE COUNT
This column lists the number of references to blocks in a buffer set. A zero in this column means that no references were made to blocks in a buffer set, a one means that one reference was made, and so on. The “=> 10” in the last row of this column means that 10 or more references were made.

NBR OF OCCURRENCES
This column shows you how many buffer sets were referenced the number of times shown in the REFERENCE COUNT column. For example, in Figure 169 on page 448, the first number in this column is 3, which says that 3 buffer sets were never referenced. The second number says that 551 buffer sets were referenced once.

PCT OF OCCURRENCES
This column shows you the value in the NBR OF OCCURRENCES column expressed as a percentage of the number of times buffer sets were reused. For example, in Figure 169 on page 448, the first number in this column is 0.46, which says that 0.46 percent of the reused buffer sets were never referenced. The second number says that 84.50 percent of the reused buffer sets were referenced once.
ACCUMUL. PCT
This column shows you the accumulated PCT OF OCCURRENCES from zero through the current reference count. For example, in Figure 169 on page 448 the third number in this column is 84.96, which says that 84.96 percent of the reused buffer sets were referenced two or less times.

DISTRIBUTION OF REFERENCES IN RANDOM SRAN CBS
This table shows you more detailed information about the ratio reported in the REFERENCES IN RANDOM SRAN CBS field. The table is similar to the DISTRIBUTION OF REFERENCES IN BUFFER SETS table, except that it shows information about references tracked by random SRAN control blocks.

REFERENCE COUNT
This column lists the number of references to blocks tracked by random SRANs. A zero in this column means that no references were made and, therefore, no random reads were issued for blocks tracked by a random SRAN. A “one” means that one reference was made to blocks tracked by a random SRAN (on the average, one reference for every tenth block tracked by the random SRAN).

NBR OF OCCURRENCES
This column shows you how many random SRANs were referenced the number of times shown in the REFERENCE COUNT column.

PCT OF OCCURRENCES
This column shows you the NBR OF OCCURRENCES expressed as a percentage of the number of times that random SRANs were reused.

ACCUMUL. PCT
This column shows you the accumulated PCT OF OCCURRENCES from zero through the current reference count.

Fields on Page C
Figure 170 shows an example of page C of the report.

---

** DEACTIVATIONS:
- NBR OF SB-DEACTIVATION 1
- NBR OF MONITORING-DEACTIVATION 0

** RESULTS OF EVALUATION OF SEQUENTIALITY:
- NBR POSITIVE RESULTS 3
- NBR NEGATIVE RESULTS 1

** RESULTS OF EVALUATION OF ACTIVITY RATE:
- NBR POSITIVE RESULTS 4
- NBR NEGATIVE RESULTS 0

** NBR RANDOM READ:
- DURING SEQUENTIAL BUFFERING PHASES 27
- DURING "MONITORING-ONLY" PHASES 659
- WHILE NOT MONITORING REFERENCE PATTERN 0

** NBR RANDOM READS WITH SEQUENTIAL REFERENCE PATTERN:
- ACCESS TO INVALID BUFFERS 25
- ACCESS AT DATA SET END 2

** NBR OF BUFFERING POSITIONS: 2,213

** INTERNAL SB-ALGORITHM VALUES:
- SDSGBPTR: BLOCKS PER TRACK 31
- SDSGBRBR: BLOCKS PER BUFSET 10
- SDSGSSCST: RELATIVE SEQ I/O COSTS 1.36
- SDSGSSINB: SIZE OF NEIGHBORHOOD 2
- SDSGSTHR1: THRESHOLD CURRENT+1 2
- SDSGSTHR2: THRESHOLD OVERLAP 3
- SDSGSTHR3: THRESHOLD NEIGHB 11

---

Figure 170. Sequential-Buffering-Detail Report Page C

Page C consists of the following sections:
DEACTIVATIONS
The fields in this section are:

**NBR OF SB DEACTIVATION**
This field shows how many times SB was deactivated.

**NBR OF MONITORING DEACTIVATION**
This field shows how many times I/O reference monitoring was deactivated.

Deactivation of I/O reference monitoring can occur for one of two reasons:
- If several consecutive periodical evaluations of the buffering process show that it is not worthwhile to use SB
- If you set a limit on the SB buffer space by specifying the MAXSB keyword in the SBONLINE control statement. The use of SB is restricted when this limit is reached.

RESULTS OF EVALUATION OF SEQUENTIALITY
This section and the next section, RESULTS OF EVALUATION OF ACTIVITY RATE, help explain why SB was deactivated (or was not activated) during the application program's execution.

The decision to activate and deactivate SB is made by a periodical evaluation of the buffering process for a particular DB-PCB/DSG control block pair. The evaluation is based on the following criteria:
- Sequentiality
- Activity rate

If the results for both tests are positive (in other words, if the results of both tests recommend use of SB), IMS will activate SB (if not active) or will continue to use SB. If at least one of the test results is negative, then IMS will deactivate SB (if active) or will continue not to use SB. After several decisions not to use SB, IMS can also deactivate monitoring of the I/O reference pattern.

The fields in this section are:

**NBR POSITIVE RESULTS**
This field shows you how many times periodical evaluation of the I/O reference pattern detected enough of a sequential reference pattern to warrant use of SB.

**NBR NEGATIVE RESULTS**
This field shows you how many times periodical evaluation of the I/O reference pattern did not detect enough of a sequential reference pattern to warrant use of SB.

RESULTS OF EVALUATION OF ACTIVITY RATE
The fields in this section are:

**NBR POSITIVE RESULTS**
This shows you how many times periodical evaluation detected an I/O activity rate high enough to warrant use of SB.

**NBR NEGATIVE RESULTS**
This shows how many times periodical evaluation determined that the I/O activity rate was not high enough to warrant use of SB.

NBR RANDOM READ
This section shows you how many random reads were issued during each of the following types of buffering phases:
**DURING SEQUENTIAL BUFFERING PHASES**
This field shows you how many random reads were issued while SB was active.

**DURING “MONITORING ONLY” PHASES**
This field shows you how many random reads were issued while SB was not active and IMS was still monitoring the I/O reference pattern.

**WHILE NOT MONITORING REFERENCE PATTERN**
This field shows you how many random reads were issued while SB was not active and IMS was not monitoring the I/O reference pattern.

**NBR RANDOM READS WITH SEQUENTIAL REFERENCE PATTERN**
This section shows you how many times random reads were issued even though the I/O reference pattern was sequential. These counters are updated only when SB is active.

The fields in this section are:

**ACCESS TO INVALID BUFFERS**
This field shows how many times a random read was issued because the contents of an SB buffer was invalid and could not be used. Some possible reasons for marking an SB buffer invalid are:

- Your IMS system is running in a block-level sharing environment. In a block-level sharing environment, more than one IMS system can read from and write to the same database. For example, if IMS system “A” reads a block into a buffer and IMS system “B” updates that block while the block is still in system “A”s” buffer, system “A”s” buffer will be marked as invalid. For more information on block-level sharing, see [IMS Version 8: Administration Guide: System](#).
- The activation and deactivation of SB during periodical evaluations marks SB buffers as invalid.
- A block was being written by another online application in the same IMS subsystem at the same time the SB buffer handler was reading it.
- The SB buffer was marked invalid because of I/O errors.

**ACCESS AT DATA SET END**
This field shows how many times the SB buffer handler issued a random read because the set of 10 blocks containing the referenced block was not completely formatted. SB never issues a sequential read at the end of a data set if the last set of consecutive blocks is not completely formatted.

**NBR OF BUFFERING POSITIONS**
This field shows how many times the SB buffer handler assumed the application program issued a DL/I call requesting a new position in the database. For example, most GU calls qualified on the key field of a root segment other than the current root segment will cause this counter to be incremented. This counter is maintained only while SB is active and IMS is monitoring the I/O reference pattern.

A high value in this field can indicate a large amount of logical random processing by the application. If other fields seem to indicate that the application did not benefit from SB, this field can explain why.

**INTERNAL SB ALGORITHM VALUES**
This section shows the values of internal counters. They are included to help IMS development during SB problem determination.
Chapter 21. Interpreting Statistical-Analysis and Log-Transaction Reports

IMS provides two utilities that extract data from IMS system logs:

- The Statistical Analysis utility produces summary reports of message activity and reports for lines and terminals. For more information see Chapter 16, “Statistical Analysis Utility (DFSISTS0),” on page 325.
- The Log Transaction Analysis utility gives detailed information of individual transaction and processing activities. For more information see Chapter 15, “Log Transaction Analysis Utility (DFSILTA0),” on page 319.

In this Chapter:
- “Statistical Analysis Utility Reports”
- “Calculating Transaction Loads” on page 456
- “Auditing Critical Transactions” on page 459
- “Log Transaction Analysis Utility Reports” on page 460
- “Examining Scheduling Activity” on page 461
- “IMS Accounting Information” on page 463

Statistical Analysis Utility Reports

The input data for the Statistics Analysis utility is a set of IMS log data sets, or user data sets created by the Log Archive utility. Each input data set can consist of multiple volumes. Several data sets can be concatenated. The utility uses two passes, each with an edit and an MVS sort operation. Multiple intermediate tape volumes are usually produced.

Restriction: You cannot use system log output from a batch system.

You can use the Transaction Analysis utility to obtain new system logs with reduced content to save processing time through the sort steps. Another savings in execution is achieved by specifying the NOTXT option on the DFSISTS0 EXEC statement. This option excludes the message text from the records that are subsequently sorted and cumulated.

Another parameter in this EXEC statement specifies the suffix value for the nucleus. If your system logs were from an execution of the online IMS system that used a different nucleus from the default 0, you must give this suffix value. Do not concatenate system log data sets from different nucleus executions.

The Statistical Analysis utility has five control statements you can use to select a subset of transaction activity.

Transaction code control statement
You can use this control statement to select a specific transaction code or groups of transaction codes. For more information see “Transaction Code Control Statement” on page 340.

Symbolic terminal name control statement
You can use this control statement to specify the LTERM name or a generic name. For example, L3270M selects all messages originating from or
directed to that LTERM. A generic name of L3270* could select L3270M and L3270B messages, the comparison being based on the characters preceding the "*.

You can further qualify the output LTERM so that only messages to a given symbolic name resulting from the input LTERM specified are selected. For more information see “Symbolic Terminal Name Control Statement” on page 340.

**Hardware terminal address control statement**

This control is similar to the symbolic terminal name control statement, except that you specify the line number and relative terminal number assigned to the physical terminal by system definition. Again, you can specify the output address as a further qualifier. If an output message was queued but not sent, it is not selected.

**Related Reading:** See “Hardware Terminal Address Control Statement” on page 341 for more information.

**VTAM terminal name control statement**

This control is similar to the symbolic terminal name control statement, except that you specify the node name for the physical terminal. Again, you can specify the output address as a further qualifier. If an output message was queued but not sent, it is not selected.

**Related Reading:** See “VTAM Terminal Name Control Statement” on page 341 for more information.

**Time control statement**

You can specify an interval as a criterion for selection. You give the start and stop times in the form YYDDD and HHMM (Julian day and clock time in minutes). This range criterion is applied to all messages selected by transaction code and terminal specifications.

**Related Reading:** See “Time Control Statement” on page 341 for more information.

If you anticipate non-printable characters in the message text, you can specify they be printed in hexadecimal format (first character above the second). Otherwise, the characters appear as blanks.

### Calculating Transaction Loads

There are two reports produced by the Statistical Analysis utility that summarize the distribution of transaction activity across a 24-hour day. Input and output message distributions are separately tabulated for each transaction code and for each device. A further report shows the response times for each transaction type expressed as percentiles. The reported data is dependent on the selection of system log data that makes up the utility input. The scope of the report can be further limited by selecting a subset of transactions and line traffic as well as the reporting interval.

**Figure 171 on page 457** shows the format of the Line and Terminal report. For each device on a line the LTERM name is given and a pair of rows of results for send and receive activity is given. The “Total Messages” column is followed by the total and average size of the messages in bytes. A series of hourly intervals divides the 24-hour day and counts of the transaction active in those intervals are recorded. Entries for devices restricted to input only or output only traffic show only one reporting line. **Figure 172 on page 457** shows the format of the Transaction report.
Calculating Transaction Loads

This organizes the data like the Line-and-Terminal report, except that it is ordered by transaction code.

<table>
<thead>
<tr>
<th>LINE RTN OR NODE</th>
<th>L I N E AND TERMINAL REPORT</th>
<th>DATE 06/17/93</th>
<th>PAGE 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/S</td>
<td>L I N E TOTAL MESSAGES</td>
<td>H OURLY DISTRIBUTION</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R E T A N AL SIZE</td>
<td>00-07</td>
<td>07-08</td>
</tr>
<tr>
<td>DSW0011 CTRL</td>
<td>13 2,098 161</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7 1,067 152</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>DSW0012 SYS</td>
<td>11 352 32</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 190 63</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>DSW0013 SY T</td>
<td>14 1,775 126</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7 1,012 144</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>DSW0014 SY T</td>
<td>15 1,151 76</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 825 165</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>DSW0015 SYS</td>
<td>12 678 56</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 491 98</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>DSW0016 SYS</td>
<td>11 355 32</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 298 74</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>DSW0017 SYS</td>
<td>10 351 35</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 190 63</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>DSW0018 SYS</td>
<td>14 1,736 124</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
</tbody>
</table>

key:

LINE RTN—Line Relative Terminal Number
R/S—Received/Sent

Figure 171. Line-and-Terminal Report

Figure 172. Transaction Report

Figure 173 on page 458 shows the format of the Transaction-Response report. This report gives the longest and shortest response times for each transaction code for the data selected as input from a set of system logs. Four columns record the percentile response times in seconds. The 25th, 50th, 75th, and 95th percentiles are given. For example, a response time within the 50th percentile is greater than or equal to 50% of the total number of response times processed for that transaction. The first line of response times given is from the completion of the receipt of the input message until the response message is successfully dequeued. In the event that an output message takes a significantly long time to be completely received at the terminal, a second line shows the receipt to the time the response message is started.
Assessing Program-to-Program Traffic

When a message processing program directs an output message to another program, that secondary transaction is queued. The transaction code is sometimes unique for convenience of the processing program's logic. Otherwise, the secondary transaction is queued along with any messages from terminal origin.

You can use the two Messages—Program-to-Program reports to separately count transaction traffic. Figure 174 illustrates the two tabulations. The column headed "Destination" appears above a list of transaction codes that were queued to another program. The originating program is not identified. The column headed "Transaction Code" appears above a list of the initial transaction codes that invoked the programs that issued the secondary transactions.

If you had program-to-program switches during conversational transaction processing, these will be included in the lists.

Obtaining Counts of Unsent Messages

The two reports titled Messages—Queued-But-Not-Sent summarize how many output messages were still in the message queues for interval covered by the input tapes. The reports are illustrated in Figure 175 on page 459. Command responses that were not sent to the terminal are indicated by (IMSSYS). The entry of NOTAVA indicates "no transaction available". This would be the case if an output message
were generated for an input not recorded in the system log input data or by a command input from the same terminal not recorded.

<table>
<thead>
<tr>
<th>MESSAGES--QUEUED BUT NOT SENT</th>
<th>TOTAL MESSAGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE 10/06/88</td>
<td></td>
</tr>
<tr>
<td>DESTINATION</td>
<td></td>
</tr>
<tr>
<td>ELEANOR</td>
<td>1</td>
</tr>
<tr>
<td>SW1050</td>
<td>1</td>
</tr>
<tr>
<td>T2741N</td>
<td>1</td>
</tr>
<tr>
<td>T2742N3</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MESSAGES--QUEUED BUT NOT SENT</th>
<th>TOTAL MESSAGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE 10/06/88</td>
<td></td>
</tr>
<tr>
<td>TRANSACTION CODE</td>
<td></td>
</tr>
<tr>
<td>(IMSSYS)</td>
<td>5</td>
</tr>
</tbody>
</table>

Figure 175. Messages—Queued-But-Not-Sent Reports

**Auditing Critical Transactions**

You can use the optional Messages report produced by the Statistical Analysis utility with the DFSIST40 program to examine the input and output data for specific transaction codes in detail. This allows you to audit exactly what was in an input message and possibly examine the output content for errors. The report is illustrated in [Figure 176 on page 460](#).

Notice the entries flagged as THIS OUTPUT NOT RESULT OF INPUT. In the figure they show several responses to IMS commands sent to line 2 terminal 1, the master terminal. The entries are flagged because the outputs do not originate from a program invoked by an input transaction. If an automated operator program is active, you can use the output to trace its activity.
You can obtain detailed data at the individual transaction level by using the Log Transaction Analysis utility. Although the data is not summarized by this utility, the detail report lines bring together many information items that help you assess the service given to a transaction type and the effect of the scheduling algorithm. The report shows actual response data because input data is the IMS log.

If you do not process the entire IMS log, data is presented from a starting checkpoint to a cutoff point. You limit the sample of transaction processing to be analyzed by specifying start time and duration in minutes, or you can give the number of checkpoints to be included after the starting checkpoint. Nonrecoverable or canceled messages are omitted.

The format of the Log-Analysis report is illustrated in Figure 177 on page 461. Times are given to the nearest tenth of a second and are elapsed times. The full list of data items for each report detail line is shown in Table 48 on page 461. You can see that the Processing Types field is a key description item.

Using the starting position and lengths of the fields in the report detail records you can specify a sort order for the second step in the utility execution.

Example: The sort control statement to cause a report to be sequenced by message class and transaction priority is:
You can also use the option of creating a DASD data set of the detail report records. Your installation could then develop an analysis program to extract and summarize the data.

### Examining Scheduling Activity

Using the data extracted by the Log Transaction Analysis utility (see Figure 177) you can examine the effect of your scheduling algorithm. Each occurrence of a transaction primarily indicates:

- Message priority and message class
- Time in input queue
- Time to process
- Time in output queue
- Total time-in-system (measured between completion of message queue input to retrieval for output)

You can look at the processing type 'S' entries to see the send and receive times. You can sort the detail report lines by transaction code and look at any critical transactions requiring rapid response time. Refer to Table 48 for details on each table line item.

#### Table 48. Log-Analysis Report Line Format

<table>
<thead>
<tr>
<th>Identification</th>
<th>Starting Position</th>
<th>Length</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence Number</td>
<td>1</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Transaction Code</td>
<td>7</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Priority of Transaction (PR)</td>
<td>16</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Class of Transaction (CL)</td>
<td>18</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
### Examining Scheduling Activity

#### Table 48. Log-Analysis Report Line Format (continued)

<table>
<thead>
<tr>
<th>Identification</th>
<th>Starting Position</th>
<th>Length</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Node Name (for VTAM)</td>
<td>22</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Input Relative Line (or VTAM node name)</td>
<td>22</td>
<td>6 (8)</td>
<td>9</td>
</tr>
<tr>
<td>Input Relative Terminal (for non-VTAM)</td>
<td>29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output Node Name (for VTAM)</td>
<td>33</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Output Relative Line (or VTAM node name)</td>
<td>33</td>
<td>6 (8)</td>
<td>9</td>
</tr>
<tr>
<td>Output Relative Terminal (for non-VTAM)</td>
<td>40</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Processing Type (PT)</td>
<td>44</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Program Name</td>
<td>46</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Dependent Region ID</td>
<td>55</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Time of SMB Enqueue (Transaction received)</td>
<td>59</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>Time of Message Schedule or GU</td>
<td>68</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>Time of CNT Enqueue (Message put on output queue)</td>
<td>77</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>Time of Program End or Next Message GU</td>
<td>86</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>Time of CNT GU (Output message starts to terminal)</td>
<td>95</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>System IDs</td>
<td>103</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Time in Input Queue</td>
<td>106</td>
<td>6</td>
<td>4, 5</td>
</tr>
<tr>
<td>Time Processing</td>
<td>113</td>
<td>6</td>
<td>6, 5</td>
</tr>
<tr>
<td>Time in Output Queue</td>
<td>120</td>
<td>6</td>
<td>7, 5</td>
</tr>
<tr>
<td>Total Time</td>
<td>127</td>
<td>6</td>
<td>8, 5</td>
</tr>
</tbody>
</table>

#### Notes to Table 48 on page 461

1. Starting position 1 is a carriage control character which alters the starting positions of the fields when producing a report on disk.

2. Processing types:
   - **A**: Program Abend or Unconnected Transaction
   - **B**: Processing restarted
   - **C**: Conversational Send/Receive Processing
   - **D**: Transmit Only Conversational Processing
   - **F**: `/FORMAT` entered (Transaction Code Field has MODNAME)
   - **M**: Message Switch
   - **O**: Region Occupancy (A region is occupied by a program that is processing transactions that existed in the input queue before the start checkpoint has encountered or a program scheduled by an unrecoverable message.)
   - **P**: Program Switch Send/Receive Processing
   - **Q**: Transmit Only Program Switch Processing
   - **R**: Program was running at time of IMS abend
   - **S**: Send/Receive Processing
Transmit Only Processing

Conversational Program Switch, Send/Receive Processing

Transmit Only Conversational Program Switch Processing

3. Time HHMMSST
4. Input queue time is from SMB enqueue to message schedule.
5. Time SSSST or OVRFLW (If the total seconds exceeds the field size, OVRFLW is printed).
6. If the wait-for-input (WFI) system option is used, the time processing field also includes the wait time between transactions.
7. Output queue time is from CNT enqueue to CNT GU.
8. Total Time is from SMB enqueue to CNT dequeue. The total time spans the complete transaction.
9. For VTAM terminals, the input relative line and input relative terminal fields are replaced with an input VTAM terminal node name. The output relative line and output relative terminal fields are replaced with an output VTAM terminal node name. The node name fields are 8 characters long.

IMS Accounting Information

The nature of accounting methods varies a great deal among data processing installations. The IMS Transaction Manager presents special difficulties, because many and varied transactions are processed by a partnership of the control region and dependent regions. Further, operationally discrete applications can be served concurrently.

For installations with IMS-dedicated processors, the overall cost of hardware and support functions is often charged back based on predicted and actual use by contributing groups. For shared systems, the processor usage can be the base for proportional cost.

Although IMS does not have an explicit accounting function, the individual events that make up the processing activity are recorded on the IMS log in considerable detail. Analysis of IMS log records can be used as a basis for charge-back algorithms. You can, for example, obtain a report from the Statistical Analysis utility of the number of transactions and the average number of DL/I calls for each transaction.

Another source of resource usage figures is the reports produced as a result of IMS Monitor data collection. Samples of processing activity can be taken on a regular basis. Accounting algorithms can use, for example, processor utilization by program.

Both of these approaches require further manipulation and calibration of the resource indicators.

If the DL/I address space option is used (LSO=S), accounting procedures based on SMF data will be affected. SMF statistics for IMS system data sets and Fast Path databases are accounted to the control region procedure. Full function databases are accounted to the DL/I address space procedure.

Using the Application-Accounting Report

The Statistical Analysis utility produces an Application-Accounting report that you can use to assess machine charges. The following breakdown is provided for each transaction and for each program:
IMS Accounting Information

- The number of messages with related total and average processor time in seconds
- The number and type of DL/I message calls
- The number and type of DL/I database calls

Figure 178 illustrates the output format.

The disadvantage is that the output is only for the interval covered by the IMS log input. You must coordinate the input data sets and accumulate the statistics. A large number of data sets would not be insignificant to process, because the utility uses two edit and sort passes, producing intermediate tape output.

**Recommendation:** If you use the utility, specify the NOTXT option on the step 1 EXEC statement, because this saves considerable sort data manipulation.

### Using IMS Transaction Profiles

You can use a composite picture of each transaction as a basis for estimating usage. The IMS Transaction profile can contain DL/I call requirements by type of call and possibly items derived from path length for other processing blocks. You can weight the message count to allow for heavy DL/I use by the transaction. Transaction statistics can be obtained on a regular basis from /DISPLAY output, for example at end-of-day or before shutdown.

The profiles should characterize the IMS workload in such a way that growth trends and major deviations from the predicted load can be traced to the transaction codes responsible.
Part 6. Appendixes
This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs
and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM’s future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM’s application programming interfaces.
Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

### Programming Interface Information

This book is intended to help database administrators and system programmers run the IMS utility programs. This book primarily documents General-use Programming Interface and Associated Guidance Information provided by IMS.

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

However, this book also documents Diagnosis, Modification, or Tuning Information which is provided to allow the customer to format and print program trace records of IMS.

**Attention:** Do not use this Diagnosis, Modification, or Tuning Information as a programming interface.

```
Diagnosis, Modification, or Tuning Information
```

Diagnosis, Modification, or Tuning Information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking: Diagnosis, Modification, or Tuning Information...

```
End of Diagnosis, Modification, or Tuning Information
```

### Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

- BookManager
- CICS
- CICS/ESA
- CICS/MVS
- DataPropagator
- DB2
- DB2 Universal Database
- DFSMSdfp
- IBM
- IMS
- IMS/ESA
- Language Environment
- Library Reader
- MVS
- MVS/DFP™
- MVS/ESA
- MVS/XA™
- NetView
- OS/390
- RACF
- Tivoli
- VTAM
- WebSphere
- z/OS

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States, other countries, or both.
UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

**Product Names**

In this book, the licensed program “DB2 Universal Database™” for OS/390 and z/OS™ is referred to as “DB2”.
Bibliography

This bibliography lists all of the information in the IMS Version 8 library.

*CICS/ESA CICS-IMS Database Control Guide, SC33-1184*
*CICS/ESA CICS-Supplied Transactions, SC33-0669*
*CICS/MVS-Supplied Transactions Manual, SC33-0240*
*DFSMSdfp Utilities, SC26-7343*
*High Level Assembler Programmer's Guide, SC26-4941-00*
*MVS Access Method Services, GC26-3841*
*MVS/DFP Linkage Editor and Loader, SC26-4564*
*MVS/DFP Utilities, SC26-4559*
*MVS/ESA Interactive Problem Control System User's Guide and Reference, GC28-1297*
*OS/VS-DOS/VSE-VM/370 Assembler Language, GC33-4010*
*Utilities for MVS/DFP, SC26-4516*

### IMS Version 8 Library

<table>
<thead>
<tr>
<th>Order Number</th>
<th>Acronym</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC27-1293</td>
<td>ADB</td>
<td>IMS Version 8: Administration Guide: Database Manager</td>
</tr>
<tr>
<td>SC27-1295</td>
<td>ATM</td>
<td>IMS Version 8: Administration Guide: Transaction Manager</td>
</tr>
<tr>
<td>SC27-1296</td>
<td>APDB</td>
<td>IMS Version 8: Application Programming: Database Manager</td>
</tr>
<tr>
<td>SC27-1300</td>
<td>APCICS</td>
<td>IMS Version 8: Application Programming: EXEC DLI Commands for CICS and IMS</td>
</tr>
<tr>
<td>SC27-1301</td>
<td>APTM</td>
<td>IMS Version 8: Application Programming: Transaction Manager</td>
</tr>
<tr>
<td>SC27-1302</td>
<td>BPE</td>
<td>IMS Version 8: Base Primitive Environment Guide and Reference</td>
</tr>
<tr>
<td>SC27-1303</td>
<td>CR</td>
<td>IMS Version 8: Command Reference</td>
</tr>
<tr>
<td>SC27-1304</td>
<td>CQS</td>
<td>IMS Version 8: Common Queue Server Guide and Reference</td>
</tr>
</tbody>
</table>

**Supplementary Publications**

<table>
<thead>
<tr>
<th>Order Number</th>
<th>Acronym</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>GC27-1299</td>
<td>LPS</td>
<td>IMS Version 8: Licensed Program Specifications</td>
</tr>
<tr>
<td>SC27-1307</td>
<td>SOC</td>
<td>IMS Version 8: Summary of Operator Commands</td>
</tr>
</tbody>
</table>
Publication Collections

<table>
<thead>
<tr>
<th>Order Number</th>
<th>Description</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>LK3T-7092</td>
<td>CD</td>
<td>IMS Version 8 Softcopy Library</td>
</tr>
<tr>
<td>LK3T-7144</td>
<td>CD</td>
<td>IMS Favorites</td>
</tr>
<tr>
<td>LBOF-7787</td>
<td>Hardcopy and CD</td>
<td>Licensed Bill of Forms (LBOF): IMS Version 8 Hardcopy and Softcopy Library</td>
</tr>
<tr>
<td>SBOF-7788</td>
<td>Hardcopy</td>
<td>Unlicensed Bill of Forms (SBOF): IMS Version 8 Unlicensed Hardcopy Library</td>
</tr>
<tr>
<td>SK2T-6700</td>
<td>CD</td>
<td>OS/390 Collection</td>
</tr>
<tr>
<td>SK3T-4270</td>
<td>CD</td>
<td>z/OS Software Products Collection</td>
</tr>
<tr>
<td>SK3T-4271</td>
<td>DVD</td>
<td>z/OS and Software Products DVD Collection</td>
</tr>
</tbody>
</table>

Accessibility Titles Cited in this Book

<table>
<thead>
<tr>
<th>Order Number</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA22-7787</td>
<td>z/OS TSO Primer</td>
</tr>
<tr>
<td>SA22-7794</td>
<td>TSO/E User’s Guide</td>
</tr>
<tr>
<td>SC34-4822</td>
<td>ISPF User’s Guide, Volume 1</td>
</tr>
</tbody>
</table>
## Index

### Special characters

//DFSSTAT report
- OSAM-Buffer-Pool Report 441
- PST-Accounting Report 439
- Sequential-Buffering-Detail Report 445
- Sequential-Buffering-Summary Report 443
- Specifying 439
- Types of Reports 439
- VSAM-Buffer-Pool report 440

### A

- ACB (application control block)
  - ACBLIB library 156
definition 155
- ACB Maintenance utility (DFSRR00)
  - ACBGEN procedure, process 157
control statements
    - BUILD 160
    - BUILD DBD 161
    - DELETE 160
    - format 160
    - requirements 159
description 156
- DFSACBCP control statement 159
examples 163
- IMS.ACBLIB 156
input 156
- JCL 157
output 156
return codes 163
- ACBGEN procedure 157
- DD statements 158
- EXEC statement 158
- ACBLIB data set 197
  - ACB Maintenance utility 156
- ACBLIB library 197
- ACCESS= parameter
  - DBD statement 22
active region messages
  - active region report fields 240
  - Log Recovery utility (DFSULTR0) 240
allocation macro
  - See Dynamic Allocation Macro (DFSMDA)
alternate PCB statement, PSB generation 115
- ALTRESP= parameter
  - PCB TYPE=TP control statement 116
- AOI (Automated Operator Interface), IOASIZE requirement 134
- Application Control Blocks (ACBs) maintaining
  - control statements 159
  - input and output 156
  - overview 155
  - restrictions 156
Application Control Blocks Maintenance utility
  - See ACB Maintenance utility

### Application-Accounting report
- example 464
- use for accounting 463
archive utility
  - See Log Archive utility (DFSUARC0)
AREA statement
  - format 44
  - keywords 45
  - parameter description 45
Automated Operator Interface (AOI)
  - See AOI (Automated Operator Interface), IOASIZE requirement

### B

- B= parameter
  - DFSERA10 OPTION statement 267
- BLKSIZE= parameter
  - DFSMDA TYPE=DFSDCMON statement 174
- BLOCK= parameter
  - DATASET statement 36
- BUFNO= parameter
  - DFSMDA TYPE=DFSDCMON statement 174
- BYTES= parameter
  - statements
    - FIELD 80
    - SEGM 57

### C

- C= keyword
  - File Select and Formatting Print utility 267
- Call Summary report
  - IMS Monitor (DB/DC) 361
  - IMS Monitor (DBCTL) 396
  - IMS Monitor (DCCTL) 418
checkpoint
  - monitoring processing effect
    - DB/DC 359
    - DBCTL 395
    - DCCTL 416
- CMPAT= parameter
  - PSBGEN statement 134
- Command Authorization user exit (DFSCCMD0) 183
- Communication Summary report
  - IMS Monitor (DB/DC) 371
  - IMS Monitor (DBCTL) 426
- Communication Wait report
  - IMS Monitor (DB/DC) 372
  - IMS Monitor (DCCTL) 427
- COMP= parameter
  - ACBGEN procedure 158
- COMPRTN= keyword
  - DEDB 68
  - DL/I 68
  - SEGM statement 67
COND= keyword
File Select and Formatting Print utility
(DFSERA10) 267

CONST= keyword
XDFLD statement 83

CONTROL statement
File Select and Formatting Print utility
(DFSERA10) 263
Control Statement Listing 85
assembler listing 86
diagnostics 85

COPY option
File Select and Formatting Print utility
(DFSERA10) 266
COPY statement
Log Archive utility 223
copying log records into user data sets, Log Archive utility 217
cross system queuing determination 380, 434

D

D= keyword
control statements
DFSERA10 CONTROL 264
DFSERA10 OPTION 268
Data Capture exit routine
See EXIT= parameter
data entry database (DEDB)
See Fast Path, DEDB
data set, dynamic allocation
See Dynamic Allocation Macro
database
monitoring buffers 369, 399
monitoring DL/I calls 356, 413
Database Buffer Pool report
IMS Monitor (DBCTL) 399
database copy
See Online Database Image Copy utility (DFSUICP0)
Database Description (DBDs)
See also DBD (Database Description) generation
generating
DEDB databases 8
GSAM databases 5
HSAM databases 4
database description rules, DBD generation 13
database Descriptions (DBDs)
generating
HDAM and PHDAM databases 6
HIDAM and PHIDAM databases 6
HISAM databases 5, 6
Index and PSINDEX databases 8
logical segment types 9
MSDBs 7
database image copy, online
See Online Database Image Copy utility
Database-Buffer-Pool report
IMS Monitor (DB/DC) 369
database, dynamic allocation
See Dynamic Allocation Macro

DATASET statement
description 30
format 32
GSAM database 33
HDAM database 34
HIDAM database 34
HISAM database 33
HSAM database 33
INDEX database 34
keywords 35
LOGICAL database 35
MSDB database 34
parameter description 35

DATXEXIT parameter
DBD statement 30

DBD (Database Description) generation
AREA statement
description 45
format 44
keywords 45
assembler listings 86
block size, specifying minimum for databases 36
coding conventions 11
control interval size, specifying minimum for databases 36
DATASET statement
description 30
dividing database into multiple data set groups 31
format 32
LABEL field 32
DBD statement 14
DBDGEN statement 85
DD statements 43, 44
description rules 13
diagnostics 85
END statement 85
error conditions 89
examples
Fast Path DEDB 99
Fast Path MSDB 98
GSAM 96
HDAM 92
HIDAM 93
HISAM 91
HSAM 90
index generation 89, 95
logical relationships 101
secondary indexes 105
secondary indexing or logical relationships 89
shared secondary indexes 107
Fast Path DEDB 8
Fast Path MSDB 7
FIELD statement
description 75
format 77
keywords 77
assembler listings 86
GSAM database 5
HDAM database 6
HIDAM database 6
HISAM database 5

474 Utilities Reference: System
DBD (Database Description) generation (continued)
HSAM database 4
index generation
  logical 9
  primary HIDAM 8
  secondary index 8
input record structure 10
LABEL field 32
LCHILD statement
  defining logical relationships 69
  defining primary index relationship 69
  defining secondary index relationships 69
  description 69
  format 72
output
  assembler listing 86
  diagnostics 85
  example 86
  load module 89
  segment flag codes 88
  types 85
overview of DBDGEN 3
  consists of 4
  control statements 3
  databases used with 4
SEGM statement
  description 46
  keyword abbreviations 56
  keywords 56
  pointer keyword options and abbreviations 58
  summary of statement types 9
XDFLD statement
  description 82
  format 82
  keywords 82
DBD generation input record structure (non-DEDB) 9
  exception 9
  requirement 10
DBD= keyword
  ACB Maintenance utility 160
DBDGEN procedure 11
  invoking 12
  JCL parameters 12
DBDGEN statement
  DBD generation 85
  FINISH statement 85
DBDGEN utility 3
DBDNAME= keyword
  DL/I PCB 119
  GSAM PCB 128
DBFULTA0 (Fast Path Log Analysis utility)
  See Log Analysis utility, Fast Path (DBFULTA0)
DBNAME= operand
  control statements
    DFSMDA TYPE=DATABASE 172
    DFSMDA TYPE=FPDDEDB 172
DC Monitor data set, dynamic allocation
  See Dynamic Allocation Macro (DFSMDA)
DD1= parameter
  AREA statement 45
DDATA= operand
  XDFLD statement 84
DDNAME= keyword
  control statements
    DFSERA10 CONTROL 264
    DFSERA10 OPTION 268
    DFSMDA TYPE=DATASET 173
    DFSMDA TYPE=DFSDCMON 173, 174
DDNOUT= keyword
  DFSERA10 CONTROL control statement 264
Deadlock Event Summary report
  IMS Monitor (DB/DC) 372
  IMS Monitor (DBCTL) 401
Deadlock reporting 275
  See also Program Isolation Trace Record Format and Print Module (DFSERA40)
  for U777 and U123 abends 275
  resultant state of the lock 279
DEDB (data entry database)
  See also Fast Path, DEDB
  defining
    areas 44
    DBD generation, description 8
    fields 77
    segments 54
  naming 21
DEDB segment edit/compression
  See COMPRTN= keyword
Detail-Listing-of-Exception-Transactions Report
  Fast Path Log Analysis utility 294
detecting bottlenecks in message processing 460
DFSACBCP control statement 159
DFSERA10 (File Select and Formatting Print utility)
  See File Select and Formatting Print utility (DFSERA10)
DFSERA30 (Record Format and Print Module)
  See Record Format and Print Module (DFSERA30)
DFSERA40
  See Program Isolation Trace Record Format and Print Module (DFSERA40)
DFSERA50 (DL/I Call Trace Data exit routine)
  See DL/I Call Trace Data exit routine (DFSERA50)
DFSIST20 (Edit Pass 2)
  Statistical Analysis utility 329
DFSIST30 (Report Writer)
  Statistical Analysis utility 329
DFSIST40 (Message Select and Copy or List)
  Statistical Analysis utility 331
DFSISTS0 (Sort and Edit Pass1)
  Statistical Analysis utility 328
DFSLTMG0 (Log Merge utility)
  See Log Merge utility (DFSLTMG0)
DFSMDA (Dynamic Allocation Macro)
  See Dynamic Allocation Macro (DFSMDA)
DFSMMNTR0, Data Communication Monitor 257
DFSMREC control statement 203
DFSOFMD0 (Offline Dump Formatter)
  See Offline Dump Formatter (DFSOFMD0)
DFSUARC0 (Log Archive utility)
  See Log Archive utility (DFSUARC0)
DFSLTR0 (Log Recovery utility)
  description 233
  modes 236

DFSUCU0 (Online Change Copy utility).
  See Online Change Copy utility (DFSUCU0)

DFSULC0 (Global Online Change utility).
  See Global Online Change utility (DFSULC0)

DFSUSVC0
  See Dynamic SVC utility (DFSUSVC0)

DFSUTR20 (IMS Monitor Report Print utility)
  See IMS-Monitor Report Print utility (DFSUTR20)

DIS statement
  Monitor Report Print utility (DFSUTR20) 258
  DISP= keyword
    DFSMDA TYPE=DATASET control statement 173
  DISPLAY command, use in accounting 464

Distribution Appendix report
  IMS Monitor (DB/DC) 375
  IMS Monitor (DBCTL) 402
  IMS Monitor (DCCTL) 429

Dividing database into multiple data set groups
  DBD generation 31

DL/I Call Trace Data exit routine (DFSERA50) 286
  File Select and Formatting Print utility
    (DFSERA10) 286

DL/I segment edit/compression
  See COMPRTN= keyword

DLI statement
  Monitor-Report Print utility (DFSUTR20) 258

DSNAME= data set
  control statements
    DFSMDA TYPE=DATASET 173
    DFSMDA TYPE=DFSDCMON 174

dual log input
  CLS mode 235
  DUP mode 235
  REP mode 235

dump format control data set
  DD statement 316
  description 317
  subset options 317

dump formatter
  See Offline Dump Formatter (DFSOFMD0)

Dynamic Allocation Macro (DFSMDMA)
  description 167
  examples 177
  Fast Path DEDBs 168
  IMS monitor data set 168
  input and output 169
  invoking the procedure 171
  JCL requirements 170
  logical relationships 169
  macro statements 172
  monitor data set 168
  multiple DEDBs 168
  OLDS 168
  output 169
  overview 167
  restrictions 169
  SLDS 168
  statement types 172

Dynamic Allocation Macro (DFSMDMA) (continued)
  DATABASE 172
  DATASET 173
  DFSDCMON 173
  FINAL 177
  FPDEDB 172
  INITIAL 172
  OLDS 175
  RECON 174
  SLDS 175
  WADS 176

Dynamic SVC utility (DFSUSVC0)
  DD statements 210
  description 209
  error processing 210
  example 211
  input 209
  JCL requirements 210
  output 209
  restrictions 209
  return codes 210

E
  E= keyword
    DFSERA10 OPTION control statement 268
  end statement 85
  END statement 85

Enhanced Select Edit exit routine (DFSERA70) 287
  error block listing (SYSPRINT)
    description of fields 237
    Log Recovery utility 237
  error ID record, interim log
    Log Recovery utility 237
  examining scheduling for critical transactions 461
  examples
    Dynamic Allocation Macro 177
    File Select and Formatting Print utility 269
    Log Archive utility 226
    selecting all log record types with token 289
    selecting specific log record types with token 290
  EXIT statement
    Log Archive utility 225
  EXIT= parameter
    DBD statement 27
    SEGM statement 65
  EXITR= keyword
    DFSERA10 OPTION control statement 268
  EXPRESS= parameter keyword
    PCB TYPE=TP control statement 117

Extended Terminal Option (ETO) 181
  extracting multiple system transaction statistics 383, 437
  EXTRIN= operand
    XDFLD statement 85

F
  Fast Path
    AREA statement
    DEDB DBD generation record 10
Fast Path (continued)

AREA statement (continued)
  description 45
  format 32
  keywords 45
area, dynamic allocation
  See Dynamic Allocation Macro

DEDB DBD generation
  description 8
  direct dependent, specifying 57
  examples 99
  input record structure 10
  sequential dependent, specifying 58

DEDB PSB generation
  alternate PCB statement 115
  positioning options (POS=) 126
  processing options (PROCOPT=) 119
  SB= keyword 125

Log Analysis utility (DFBULTA0) 291
  log intervals 291
MSDB DBD generation
  description 7
  examples 98
MSDB PSB generation
  alternate PCB statement 115
  examples 140
  processing options (PROCOPT=) 119
  PSB PROCOPT= parameter 121

Field statement
  DEDB database 77
  HDAM and PHDAM database 76
  HIDAM and PHIDAM database 76
  HISAM database 76
  HSAM database 76
  Index database 77
  MSDB database 77
FIELD statement 75
  description 75
  format 77
  keywords 77

File Select and Formatting Print utility (DFSERA10)
  COND= keyword 267
  control statements 281
  COMMENTS 269
  CONTROL 263
  description 263
  END 269
  OPTION 264
  COPY option 266
  DDNAME= keyword 268
  description 261
  description DD statements 262
  DL/I Call Trace Data exit module (DFSERA50) 286
Enhanced Select Edit exit routine (DFSERA70) 287
  examples 269
  FLDLEN= keyword 267
  FLDTYP= keyword 266
  input 261
  Intent Failure 358
  JCL requirements 262
  description 262

File Select and Formatting Print utility (DFSERA10) (continued)
  JCL requirements (continued)
    examples 269, 273
  NEGOF option 266
  OFFSET= keyword 266
  OPTION statement
    PARM= parameter, subparameters of 287
    optional keywords 266
    output 261
    overview 261
    PARM= keyword 266
  PRINT option 265
  Program Isolation (PI) Trace Record Format and
  Print Module (DFSERA40)
    control statements 283
    description 282
    sample 283
  PRTSYS= keyword 269
  Record Format and Print Module (DFSERA30)
    control statements 281
    description 275
  STARTAF= keyword 267
  STOPAFT= keyword 267
  SYM= keyword 266
  Trace Table Record Format and Print Module
  (DFSERA60) 287
  VALUE= keyword 266

File Select and Formatting Print utility
  (DFSERA10)EXITR= keyword 268
FINISH statement 85
  FLDLEN= keyword
  DFSERA10 OPTION control statement 267
  FLDTYP= keyword
  DFSERA10 OPTION control statement 266
forced EOV
  Log Archive utility 217
  FORMAT library 197
  FORMATA, INITMOD procedure 203
  FREQ= operand
  SEGM statement 58
  FRSPC= keyword
  DATASET statement 41

G
General Iwait Time Events report
  IMS Monitor (DB/DC) 368
General IWAIT Time Events report
  IMS Monitor (DCCTL) 425
Generalized Sequential Access Method
  See GSAM (Generalized Sequential Access Method)
Global Online Change Utility (DFSUOLC0)

Index 477
GSAM (Generalized Sequential Access Method) (continued)
  DBD generation
    example 96
    specification 5, 23
  PCB generation
    example 138

H
  H= statement
    DFSERA10 OPTION control statement 267
  HDAM database
    See DBD (Database Description) generation
  HIDAM database
    See DBD (Database Description) generation
  hiprobespace buffers, VSAM Buffer Pool
    //DFSSTAT 441
  IMS Monitor (DB/DC) 370
  IMS Monitor (DBCTL) 400
  HISAM database
    See DBD (Database Description) generation
  HSAM database
    See DBD (Database Description) generation

I
  IMS Monitor Report Print Program and MSC reports
    interpreting for DB/DC 434
    interpreting for DCCTL 380
  IMS Monitor Report Print utility (DFSUTR20)
    analysis control data set 258
    description 257
    JCL example 259
    JCL requirements 257
    statements
      DIS 258
      DLI 258
      ONLY DLI 258
  IMS Monitor Reports
    Buffer Pool Statistics 352
    Call Summary 361
    Communication Summary 371
    Communication-Wait 372
    Database Buffer Pool 369
    Deadlock Event Summary 372
    Distribution-Appendix report 375
    General Iwait Time Events 368
    Intent Failure 358
    Latch Conflict Statistics 374
    Line Functions 371
    Log Archive utility (DFSUARCO) 215
    Message Format Buffer Pool 366
    Message Queue Pool 367
    MSC Queuing Summary 382
    MSC Summaries 381
    MSC Traffic 380
    Offline Dump Formatter utility (DFSOFMD0) 313
  IMS Monitor Reports (continued)
    output selection options 354
    overview 351
    Pool Space Failure Summary 372
    Program I/O 362
    Program Summary 360
    Programs by Region 355
    Region and Jobname 353
    Region Summary 355
    Region Wait 355
    Run Profile 353
    System Configuration 352
    verifying report occurrences 354
    VSAM Buffer Pool 369
  IMS Monitor Reports (DBCTL)
    adding to 388
    Call Summary 396
    Deadlock Event Summary 401
    Distribution Appendix report 402
    Intent Failure 394
    Latch Conflict Statistics 401
    output selection options 390
    overview 387
    Pool Space Failure Summary 401
    Program I/O 397
    Program Summary 396
    Programs by Region 391
    Region Summary 390
    Region Wait 391
    Run Profile 389
    System Configuration 388
    verifying report occurrences 390
    VSAM Buffer Pool 400
  IMS Monitor Reports (DCCTL)
    Call Summary 418
    Communication Summary 426
    Communication Wait 427
    Distribution Appendix report 429
    General Iwait Time Events 425
    Latch Conflict Statistics 428
    Line Functions 426
    Message Format Buffer Pool 423
    Message Queue Pool 424
    MSC Queuing Summary 437
    MSC Summaries 436
    MSC Traffic 434
    output selection options 411
    overview 407, 408
    Pool Space Failure Summary 428
    Program I/O 419
    Program Summary 417
    Programs by Region 412
    Region and Jobname 411
    Region Summary 412
    Region Wait 412
    Run Profile 410
    System Configuration 410
    verifying report occurrences 411
  IMS Monitor Reports for DB/DC
    Log Merge utility (DFSLTMG0) 229
    Log Recovery utility (DFSULTR0) 233
IMS Monitor Reports for DB/DC  (continued)
  Log Transaction Analysis utility (DFSILTA0) 319
  Security Maintenance utility (DFSISMPO) 179
  Statistical Analysis utility (DFSISTS0) 325
  Transaction Queueing 368
  VSAM Buffer Pool 407
IMS Monitor Reports for DBCTL
  Database Buffer Pool 407
IMS Monitor timed events
  checkpointing 349
  description 347
  DL/I call NOT-WAIT times
    DB/DC 350
    DCCTL 408
during message input 348, 349
  elapsed execution 349
  idle for intent 349
  NOT-WAIT time 408
  schedule of first DL/I call 349
  scheduling and termination 349
summary
  DBCTL 387
  DCCTL 408
trace intervals
  DB/DC 352
  DBCTL 388
  DCCTL 410
wait time 349
  wait-for-input (WFI)
    DB/DC 349, 366
  DCCTL 423
IMS-issued subsystem detected deadlocks 281
IMSACBA, INITMOD procedure 203
IMSDALOC procedure, process 170
index database
  See DBD (Database Description) generation
INDEX DBD generation
  logical DBD 9
  overview 8
  primary HIDAM index 8
  secondary index 8
INDEX= operand
  DBDGEN LCHILD statement 74
INDICES= operand
  SENSEG statement 130
INITMOD procedure
  DFSMREC control statement 203
  FORMATA 203
  IMSACBA 203
  JCL 202
  MODBLKSA 203
  MODSTAT record 203
  PROC statement 202
Intent Failure Summary Report
  IMS Monitor (DB/DC) 358
  IMS Monitor (DBCTL) 394
Interactive Dump Formatter 313
interim log error ID record
  Log Recovery utility 237
interpreting (continued)
  IMS Monitor Reports
    DBCTL 385
    DCCTL 407
    MSC 434
  Transaction Analysis Report 383
  Transaction Analysis Reports 455
IOASIZE= parameter
  PSBGEN statement 134
IOEROPN= parameter
  PSBGEN statement 135
IPCS (Interactive Problem Control System)
  Interactive Dump Formatter 313
  Offline Dump Formatter 315
  Offline Dump Formatter, user control statement 315
K
keyword
  See individual keyword listings
L
L= keyword
  DFSERA10 OPTION control statement 267
LABEL field
  alternate PCB statement 116
  DBD generation 32
  DL/I PCB statement 118
  GSAM PCB statement 128
  LANG= parameter
    PSBGEN statement 133
Latch Conflict Statistics report
  IMS Monitor (DB/DC) 374
  IMS Monitor (DBCTL) 401
  IMS Monitor (DCCTL) 428
LCHILD statement
  abbreviations 72
  HDAM and PHDAM databases 70
  HIDAM and PHIDAM databases 71
  HIDAM Primary index relationship 69
  HIDAM Primary Index relationship 69
  HISAM databases 70
  INDEX databases 72
  keywords 72
  logical relationships 69
  PSINDEX databases 72
  secondary index relationship 69
  Secondary Index relationship 69
Line Functions report
  IMS Monitor (DCCTL) 426
Line-and-Terminal statistics report 457
Line-Functions report
  IMS Monitor (DB/DC) 371
link queuing time assessments 382, 437
LIST= operand
  alternate PCB statement 117
  GSAM PCB statement 128
  PSBGEN statement 127
LOCKMAX= parameter
  PSBGEN statement 135, 136
Log Analysis utility
  Fast Path (DBFULTA0)  291
Log Archive utility (DFSUARC0)
  Batch DASD SLDS archive  216
  control statements  222
  COPY statement  223
  copying log records into user data sets  217
  creating an RLDS  216
  DD Statements  220
  description  215
  error processing  225
  examples  226
  EXIT statement  225
  IMSPLEX parameter  220
  JCL requirements  220
  OLDS archive  215
  OLDS input  217
  omitting log records on SLDS  217
  optional functions  216
  program output  218
  RLDS (Recovery Log Data Set)  216
  SLDS input  218
  SLDS statement  222
  specifying forced end of volume  217
  specifying user exit routines  217
  log error ID record, interim
    Log Recovery utility  237
Log Merge utility (DFSULTMG0)
  control of log output  383, 437
  control statement format  229
  controlling the log merge  229
  coordinating MSC logs  383, 437
  DD Statements  231
  description  229
  JCL requirements  231
  MSC (Multiple Systems Coupling)  229
  program inputs  229
  program outputs  231
  restriction  229
  sample control statement  231
LOG parameter
  DBD statement  29
  SEGM statement  67
log record
  Statistical Analysis utility  326
Log Recovery utility (DFSULTR0)
  active region messages  240
  CLS mode  233
  CLS mode error listing  237
  Control Statements  243
  creating a new log  245
  creating an interim log  244
  DD statements  241
  description  233
  dual log input  235
  Dump of data records  239
  DUP mode  233
  DUP mode error listing  237
  DUP mode, warning  233
  error block listing (SYSPRINT)  237
  error processing  246
Log Recovery utility (DFSULTR0) (continued)
  examples  247
  input  234
  interim log error ID record  237
  JCL requirements  241
  modes  233, 236
  OLDS recovery  234
  output  236
  print active PSB reports  246
  PSB mode  233
  REP mode  234
  REP mode verification messages  239
  single log input  234
  SLDS recovery  234
Log Transaction Analysis utility (DFSILTA0)
  description  319
  ID column for MSC entries  438
  MSC statistics  383, 437
  parameter descriptions  320
  program inputs  320
  program outputs  320
  reports produced
    description  460
    Log-Analysis report  460
LOGICAL operand
  DATASET statement  35
  LTERM= operand
    PCB TYPE=TP parameter  116
  LU 6.2  181
M
main storage database (MSDB)
  See Fast Path, MSDB
making changes online
  Global Online Change Copy utility  204
  Online Change Copy utility  197
MATRIX library  197
MAXQ= parameter keyword
  PSBGEN statement  134
MBR= parameter
  DBD generation  12
  PSB generation  113
merging logs for MSC  383, 437
Message Format Buffer Pool Report
  IMS Monitor (DB/DC)  366
  IMS Monitor (DCCTL)  423
Message Queue Pool Report
  IMS Monitor (DB/DC)  367
Message Queue Pool Report
  IMS Monitor (DCCTL)  424
Message Queue Pool Report
  IMS Monitor (DB/DC)  367
Messages report
  example  459
Messages—Program-To-Program Report
  example  458
Messages—Queued-But-Not-Sent Report
  example  459
MODBLKSL library  197
MODBLKSA, INITMOD procedure  203
MODEL= parameter
  DATASET statement  35
MODIFY= operand
   PCB TYPE=TP parameter 116
MODSTAT record, INITMOD procedure 203
monitor data set, dynamic allocation
   See Dynamic Allocation Macro (DFSMDA)
Monitor Report Print program.
   See IMS Monitor Report Print utility
monitor trace interval
   IMS Monitor Report (DB/DC) 352
   IMS Monitor Report (DBCTL) 388
   IMS Monitor Report (DCCTL) 410
monitoring
   application program elapsed time
      DCCTL 417
database buffers
   DB/DC 369
   DBCTL 399
dependent regions
   DB/DC 355
   DBCTL 390
   DCCTL 412
I/O for application program DL/I calls
   DCCTL 419
internal resource usage
   DB/DC 372
   DBCTL 401
   DCCTL 428
line activity
   DB/DC 371
   DCCTL 426
message handling
   DB/DC 372
   DCCTL 427
message queue handling
   DB/DC 362, 367
   DBCTL 395, 397
   DCCTL 424
MFS activity
   DB/DC 366
   DCCTL 423
using frequency distribution
   DB/DC 375
   DBCTL 402
   DCCTL 429
MSC (Multiple Systems Coupling)
   control of log output 383, 437
cross system queuing determination 434
cross-system queuing determination 380
IMS Monitor Report Print Program 380, 434
Interpreting Distribution Appendix Output 433
Interpreting IMS Monitor MSC Report 380, 434
Log Analysis Report
   ID column for MSC entries 438
   use for MSC transactions 438
Log Merge utility
   coordinating MSC logs 383, 437
   input 229
   output 229
Log Transaction Analysis utility (DFSILTA0) 383, 437
MSC (Multiple Systems Coupling) (continued)
   Log-Analysis Report
      ID column for MSC entries 384
      use for MSC transactions 383
MCS-Queuing Report
   interpreting 380
   merging logs for MSC 383, 437
MSC-Queuing Report
   assessing link queuing times 382, 437
   example 383, 437
   interpreting 434
MSC-Summarizes Report
   assessing queue sizes 381, 436
   content 381, 436
   example 382, 436
   interpreting 380, 434
MSC-Traffic Report
   content 380, 434
determining cross system queuing 434
determining cross-system queuing 380
   example 381, 435
   transaction statistics 383, 437
MSC Summaries Report
   IMS Monitor (DB/DC) 380
   IMS Monitor (DCCTL) 434
MSDB DBD generation, description 7

N
NAME= parameter
   statements
      alternate PCB statement 116
      DBD 22
      DL/I PCB 119
      FIELD 77
      GSAM PCB 128
      LCHILD 72
      SEGM 57
      SENFLD 132
      SENSEG 129
      XDFLD 83
NEGOF option
   File Select and Formatting Program
      (DFSERA10) 266
NOLOG parameter
   DBD statement 29
   SEGM statement 67
not message-driven option
   Fast Path Log Analysis utility 309
NULLVAL= operand
   XDFLD statement 84

O
O= keyword
   control statements
      DFSERA10 CONTROL 264
      DFSERA10 OPTION 266
Offline Dump Formatter (DFSOFMID0)
   description 313
Offline Dump Formatter (DFSOFMD0) (continued)
dump format control data set
  DD statement 316
description 317
subset options 317
environments
  DB batch 315
  DB/DC 314
  DBCTL 314
  DCCTL 314
  TM batch 315
input 315
IPCS 315
load modules 314
migration considerations 314
output 315
restrictions 314
SDUMP 314
OFFSET= keyword
DFSERA10 OPTION control statement 266
OLCSTAT data set
description 204
initializing 204
recover procedure 204
OLCUTL procedure, process 198
OLDS (online log data set)
and recover points 218
archive 215
dual OLDSs 217
input to Log Archive utility 217
recovery using the Log Recovery utility 234
termination 218
OLIC= operand
  PSBGEN statement 135
omitting log records on SLDS
  Log Archive utility 217
Online Change Copy utility (DFSUOCU0)
active library 197
cancellation 198
DD statements 200
description 197
DFSMREC control statement 203
EXEC statement 200
inactive library 197
INITMOD procedure 202
INITMOD procedure, JCL 202
JCL 201
libraries used 197
MSDB 198
OLCUTL procedure 198
PROC statement 199, 202
requirements 197
restrictions 198
staging library 197
Online Change utilities
  Global Online Change 197
  Online Change Copy 197
Online Database Image Copy utility (DFSUICP0)
PSBGEN specifications required 114, 121
ONLY DLI statement, Monitor Report Print utility
  (DFSUTR20) 258
OSAM data sets block size 39
OSAM-Buffer-Pool Report 441
output sequence and information from IMS Monitor
  Report
  DB/DC 352
  DBCTL 387
  DCCTL 409
Overall Summary of Resource Usage and Contentions
  for All Transaction Codes and PSBs Report
  Fast Path Log Analysis utility 300
Overall Summary of Transit Times by Transaction Code
  for IFP Regions Report
  Fast Path Log Analysis utility 300
OVFLW= operand
  DATASET statement 36

P
  P= keyword
  DFSERA10 OPTION control statement 269
PAIR= keyword
  LCHILD statement 74
PARENT= operand
  SEGM statement 57
  SENSEG statement 130
PARM= keyword
  DFSERA10 OPTION control statement 266
  subparameters of
    TOKEN= subparameter 288
    XFMT= subparameter 287
Partitioned Hierarchical Direct Access Method
  (PHDAM) 95
Partitioned Hierarchical Indexed Direct Access Method
  (PHIDAM) 96
PASSWD= parameter
  DBD statement 26
password security 182
/LOCK and /UNLOCK command 182
PCB statement
database PCB size 117
DL/I or Fast Path database 117
GSAM 128
SENSEG statement 129
PCBNAME= parameter
  alternate PCB statement 117
  DL/I PCB statement 119
  GSAM PCB statement 128
PHDAM (Partitioned Hierarchical Direct Access Method)
  DBD generation
  example 95
PHIDAM (Partitioned Hierarchical Indexed Direct Access Method)
  DBD generation
  example 96
POINTER= operand
  LCHILD statement 73
  SEGM statement 58, 60
Pool Space Failure Summary report
  IMS Monitor (DBCTL) 401
  IMS Monitor (DCCTL) 428
Pool-Space-Failure-Summary report
IMS Monitor (DB/DC)  372
POS= parameter keyword
PCB TYPE=DB parameter  126
PRINT option
File Select and Formatting Print utility
(DFSERA10)  265
PROC statement  11
procedure
ACBGEN  157
DBDGEN  11
IMSDALOC  170
INITMOD  202
OLCUTL  198
PSBGEN  113
Security  184
SECURITY  184
PROCOPT= operand
Fast Path  121
PCB
  type=DB  119
  type=GSAM  128
SENSEG statement  130
PROCSEQ= operand
PCB TYPE=DB statement  127
Program I/O report
IMS Monitor (DBCTL)  397
IMS Monitor (DCCTL)  419
Program Isolation (PI) Trace Record Format and Print Module (DFSERA40)
control statements  283
output sample  283
program output
Log Archive utility  218
Program Summary Report
IMS Monitor (DB/DC)  360
IMS Monitor (DBCTL)  396
IMS Monitor (DCCTL)  417
Program-I/O Report
IMS Monitor (DB/DC)  362
Programs by Region Report
IMS Monitor (DB/DC)  355
IMS Monitor (DBCTL)  391
IMS Monitor (DCCTL)  412
PRTSYS= keyword
DFSERA10 OPTION  269
PSB (Program Specification Block) (continued)
examples (continued)
GSAM  138
  logical database  142
  sample hierarchic data structure  137
  shared secondary index  150
type=GSAM  128
execution  114
generating
  control statement formats  162
  control statements  115
  input and output  111
  overview  111
  specifying options for DEDBs  129, 132, 136
output
  assembly listing  137
  control statement listing  136
  diagnostics  136
  error conditions  137
  load module  137
  PCBNAME = parameter  139
  PCBs (Program Communication Blocks)  111
  Requirements  111
  Rules  111
  six input/output statement types  111
PSB= operand
ACB Maintenance utility  160
PSBGEN procedure, process  113
PSBGEN statement
  LANG= parameter  133
  maximum number of database PCBs  117
  PSB generation  132
PSBNAME= parameter
PSBGEN statement  133
PST-Accounting Report  439
PTR= keyword
LCHILD statement  73
SEGM statement  60

R

RACF
See Resource access control facility (RACF)
Recapitulation-of-the-Analysis Report
Fast Path Log Analysis utility  304
RECFM= operand
DATASET statement  41
Record Format and Print Module (DFSERS30)
additional information gathered  280
Deadlock report  275
File Select and Formatting Print utility
(DFSERA10)  275
lock states  278
reporting anomaly  280
resultant state of the lock  279
Selecting only the Deadlock block  281
special situations  280
subsystem detected deadlocks  281
RECORD= operand
DATASET statement  41

Index  483
Recovery utility

See Database Recovery Control utility

Region and Jobname Report
IMS Monitor (DB/DC) 353
IMS Monitor (DBCTL) 390
IMS Monitor (DCCTL) 412

Region Summary Report
IMS Monitor (DB/DC) 355
IMS Monitor (DBCTL) 391
IMS Monitor (DCCTL) 412

Region Wait Report
IMS Monitor (DB/DC) 355
IMS Monitor (DBCTL) 391
IMS Monitor (DCCTL) 412

REL= keyword
DATASET statement 42
REPL= parameter keyword
SENFLD statement 132
REPLACE= parameter keyword
SENFLD statement 132

Resource access control facility (RACF)
uses with Security Maintenance utility 183
resource access security, IMS 182

RGN= parameter
procedures
ACBGEN 158
DBDGEN 12
PSBGEN 114

RLDS (recovery log data set)
output to Log Archive utility 218

RLDS (Recovery log data set) 216

RMNAME= operand
DBD statement 25

ROOT= parameter
AREA statement 45

RULES= keyword
LCHILD statement 74
SEGM statement 62

Run Profile Report 352, 409
adding generalized processing ratios
DB/DC 354
DBCTL 389
DCCTL 411
IMS Monitor (DB/DC) 353
IMS Monitor (DBCTL) 389
IMS Monitor (DCCTL) 410

S

SAMETRM= parameter keyword
PCB TYPE=TP parameter 116
SB-Detail report 445
SB-Summary Report 443
SB= keyword
PCB TYPE=DB parameter 125
SCAN= operand
DATASET statement 41

SDUMP
Offline Dump Formatter 314
SEARCHA= keyword
DATASET statement 42
-secondary index DBD generation 8

secondary index relationships 69
SECURITY macro statement
resource access security 182

Security Maintenance utility (DFSISMP0)
command authorization user exit 183
control statements 188
description 179, 188
execution 188
IMS resource access security 182
input 180
input statement operands 188
invoking the procedure 188
JCL requirements 186
LTERM security 182
output 180, 190
overview 179
password security 182
Resource access control facility (RACF) 183
restrictions 181
security options 179, 181
sign-on verification 183
transaction command security 182

SECURITY procedure, process 184

SEGM statement
DEDB database 54
description 46
format 56
HDAM database 48
HIDAM database 51
HISAM database 47
HSAM database 46
INDEX database 56
keyword abbreviations 56
MSDB database 54
PHDAM database 50
PHIDAM database 53
pointer keyword options and abbreviations 58, 59
PSINDEX database 56

SEGMENT= operand
XDFLD statement 83
selecting
extended log formatting for X’50’ log records
XFMT= subparameter 287
log records by recovery token
example of selecting all record types 289
delete example of selecting specific record types 290
TOKEN= subparameter 288

SENSEG statement
maximum number 129
PROCOPT option 130
PSB generation 129
SF=parameter
INITMOD 203
sign-on verification security 183
/SIGN command 183
single log input
Log Recovery utility 234
SIZE= keyword
AREA statement 45
DATASET statement 38
SKIP= keyword
  File Select and Formatting Print utility
    (DFSERA10) 264
SLDS (system log data set)
  batch archive 216
  closing 233
  DUP mode 233
  input to Log Archive utility 218
  omitting log records on 217
  output to Log Archive utility 218
  recovery using the Log Recovery utility 234
SLDS (system log data set) statement
  Log Archive utility 222
SOURCE= keyword
  SEGM statement 63
SOUT keyword
  procedures
    OLCUTL 199
    SECURITY 186
SOUT= parameter
  procedures
    ACBGEN 158
    DBDGEN 12
    IMSDALOC 170
    INITMOD 203
    PSBGEN 113
Specifying DBD generation
  GSAM database 5
  HIDAM and PHIDAM database 6
  HISAM database 5
  HSAM database 4, 8
  Index and PSINDEX databases 8
  Logical database 9
  MSDB database
    header information(BHDR) 7
    secondary index database 8
SRCH= keyword
  XDFLD statement 84
SSASIZE= parameter keyword
  PSBGEN statement 134
SSPTR= keyword
  SEGM statement 65
  SENSEG statement 130
START= keyword
  FIELD statement 80
  SENFLD statement 132
STARTAF= statement
  DFSERA10 OPTION control statement 267
Statistical Analysis utility (DFSISTS0) (continued)
  program modules
    EDIT PASS2 (DFSIST20) 329
    Message Select and Copy or List
      (DFSIST40) 331
    Report Writer (DFSIST30) 329
    SORT and EDIT PASS1 (DFSISTS0) 328
  reports produced, descriptions and examples
    Application-Accounting report 330, 333
    description 455
    IMS-Accounting report 330, 333
    Line-and-Terminal report 329, 332
    messages produced by Message Select and Copy
      (DF) 334, 340
    Messages Queued But Not Sent (by
destination) 329, 331
    Messages Queued But Not Sent (by transaction
code) 329, 332
    Messages, Program-to-Program (by
destination) 329, 332
    Messages, Program-to-Program (by transaction
code) 329, 332
    Transaction report 330, 333
    Transaction-Response report 330, 333
  selecting a subset of transactions 455
  transaction profiles 464
  utility control statements
    descriptions 340
    hardware terminal address 341
    nonprintable character 342
    symbolic terminal name 340
    time 341
    transaction code 340
    VTAM terminal name 341
  utilization for accounting 463
STOPAFT= statement
  DFSERA10 OPTION control statement 267
  File Select and Formatting Print utility
    (DFSERA10) 264
SUBSEQ= operand
  XDFLD statement 84
subset pointers
  SENSEG statement 130
Summary-of-Exception-Detail-by-Transaction-Code (for
  IFP Regions) Report
  Fast Path Log Analysis utility 299
Summary-of-Region-Occupancy Report
  Fast Path Log Analysis utility 302
Summary-of-VSO-Activity Report
  Fast Path Log Analysis utility 303
SVC utility
  See Dynamic SVC utility (DFSUSVC0)
SYM= option
  DFSERA10 OPTION control statement 266
SYS= keyword
  OLCUTL procedure 199
SYS= parameter
  INITMOD procedure 202
SYS2= keyword
  procedures
    IMSDALOC 170
SYS2= keyword (continued)

SECURITY 186
SYS2= parameter

procedures

ACBGEN 158
DBDGEN 13
INITMOD 202
PSBGEN 114

SYSMDUMP

Offline Dump Formatter 315
SYSPRINT

Log Archive utility 218
System Configuration Report

IMS Monitor (DB/DC) 352
IMS Monitor (DBCTL) 388
IMS Monitor (DCCTL) 410

T

T= keyword

DFSERA10 OPTION control statement 266
TOKEN= subparameter 288
Trace Table Record Format and Print Module
(DFSERA60) 287
transaction command security 182
transaction flow and IMS Monitor events 347
transaction reports

See Statistical-Analysis utility
transaction statistics in MSC 383, 437
Transaction-Response report
description 457
example 457
Statistical Analysis utility (DFSISTS0)
reports 330, 333
Transaction-Statistics report 457
TYPE= parameter
alternate PCB statement 116
DATASET 173
DFSDCMON 173
DFSMDA 172
dl/i PCB statement 119
FIELD statement 80
FINAL 177
FPDEDB 172
GSAM PCB statement 128
INITIAL 172
OLDS 175
RECON 174
SEGM statement 58
SLDS 175
TP 116

U

UNIT= keyword

DFSMDA TYPE=DFSCMON statement 174
UOW= parameter

AREA statement 45
utilities

ACBGEN 155
DBDGEN 3

Utilities (continued)

consists of 4
count statements 3
databases used with 4
Dynamic Allocation 167
Dynamic SVC utility (DFSUSVC0) 209
Fast Path Log Analysis utility (DBFULTA0) 291
File Select and Formatting Print utility

(DFSERA10) 261
Global Online Change utility (DFSUOLC0) 204
IMS Monitor Report Print utility (DFSUTR20) 257
Interpreting DFSSTAT Reports 439
Interpreting IMS Monitor Reports 347
Interpreting IMS Monitor Reports for DBCTL 385
Interpreting IMS Monitor Reports for DCCTL 407
Interpreting-Statistical-Analysis-and-Log-Transaction
Reports 455
Log Archive utility (DFSUARC0) 215
Log Merge utility (DFSLTMG0) 229
Log Recovery utility (DFSULTR0) 233
Log Transaction Analysis utility (DFSILTA0) 319
Offline Dump Formatter utility (DFSOFMD0) 313
Online Change Copy utility (DFSUOCU0) 197
PSBGEN 111
Security Maintenance utility (DFSISMP0) 179
Statistical Analysis utility (DFSISTS0) 325

V

V= keyword

DFSERA10 OPTION control statement 266
VALUE= keyword

DFSERA10 OPTION control statement 266
VERSION parameter

DBD statement 29
VIEW=MSDB operand

PSBGEN statement 127
VSAM Buffer Pool Report

/DFSSTAT 439
description 440
IMS Monitor (DB/DC) 369
IMS Monitor (DBCTL) 400

W

WADS (write-ahead data set)

CLS mode 233, 240
data set 242
NOWADS 244
wait-for-input (WFI)

exclusion from program summary report 360
on program I/O report 349
program I/O report 423
program-I/O report 366
time between transactions 463

X

XDFLD statement 82
description 82
format 82
XDFLD statement (continued)
  HDAM database  82
  HISAM database  82
  keywords  82
  parameter description  83
  PHDAM database  83
  XFMT= subparameter  287