Fast Path Offline Suite
User Guide

Supporting

Version 3.10 of Fast Path Analyzer/EP
Version 3.10 of Fast Path Reorg/EP

December 2011
Contacting BMC Software

You can access the BMC Software website at http://www.bmc.com. From this website, you can obtain information about the company, its products, corporate offices, special events, and career opportunities.

United States and Canada

<table>
<thead>
<tr>
<th>Address</th>
<th>Telephone</th>
<th>Fax</th>
</tr>
</thead>
</table>
| BMC SOFTWARE INC
2101 CITYWEST BLVD
HOUSTON TX 77042-2827
USA | 1 713 918 8800 or 1 800 841 2031 | 1 713 918 8000 |

Outside United States and Canada

<table>
<thead>
<tr>
<th>Telephone</th>
<th>Fax</th>
</tr>
</thead>
<tbody>
<tr>
<td>+01 713 918 8800</td>
<td>+01 713 918 8000</td>
</tr>
</tbody>
</table>

© Copyright 2000-2011 BMC Software, Inc.

BMC, BMC Software, and the BMC Software logo are the exclusive properties of BMC Software, Inc., are registered with the U.S. Patent and Trademark Office, and may be registered or pending registration in other countries. All other BMC trademarks, service marks, and logos may be registered or pending registration in the U.S. or in other countries. All other trademarks or registered trademarks are the property of their respective owners.

IBM, IMS, MVS, OS/390, Redbooks, z/Architecture, and z/OS are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

The information included in this documentation is the proprietary and confidential information of BMC Software, Inc., its affiliates, or licensors. Your use of this information is subject to the terms and conditions of the applicable End User License agreement for the product and to the proprietary and restricted rights notices included in the product documentation.

Restricted rights legend

U.S. Government Restricted Rights to Computer Software. UNPUBLISHED -- RIGHTS RESERVED UNDER THE COPYRIGHT LAWS OF THE UNITED STATES. Use, duplication, or disclosure of any data and computer software by the U.S. Government is subject to restrictions, as applicable, set forth in FAR Section 52.227-14, DFARS 252.227-7013, DFARS 252.227-7014, DFARS 252.227-7015, and DFARS 252.227-7025, as amended from time to time. Contractor/Manufacturer is BMC SOFTWARE INC, 2101 CITYWEST BLVD, HOUSTON TX 77042-2827, USA. Any contract notices should be sent to this address.
Customer support

You can obtain technical support by using the BMC Support Central website or by contacting Customer Support by telephone or e-mail. To expedite your inquiry, see “Before contacting BMC.”

Support website

You can obtain technical support from BMC 24 hours a day, 7 days a week at http://www.bmc.com/support. From this website, you can

- read overviews about support services and programs that BMC offers
- find the most current information about BMC products
- search a database for issues similar to yours and possible solutions
- order or download product documentation
- download products and maintenance
- report an issue or ask a question
- subscribe to receive proactive e-mail alerts when new product notices are released
- find worldwide BMC support center locations and contact information, including e-mail addresses, fax numbers, and telephone numbers

Support by telephone or e-mail

In the United States and Canada, if you need technical support and do not have access to the web, call 1 800 537 1813 or send an e-mail message to customer_support@bmc.com. (In the subject line, enter SupID:<yourSupportContractID>, such as SupID:12345). Outside the United States and Canada, contact your local support center for assistance.

Before contacting BMC

Have the following information available so that Customer Support can begin working on your issue immediately:

- product information
  - product name
  - product version (release number)
  - license number and password (trial or permanent)
- operating system and environment information
  - machine type
  - operating system type, version, and service pack or other maintenance level such as PUT or PTF
  - system hardware configuration
  - serial numbers
  - related software (database, application, and communication) including type, version, and service pack or maintenance level
- sequence of events leading to the issue
- commands and options that you used
- messages received (and the time and date that you received them)
  - product error messages
  - messages from the operating system, such as file system full
  - messages from related software
License key and password information

If you have questions about your license key or password, use one of the following methods to get assistance:

- Send an e-mail message to customer_support@bmc.com.
Contents

About this book 29
  Related publications .................................................. 30
  Conventions .................................................................. 31
  Syntax statements ..................................................... 31
  Summary of changes ................................................ 32

Chapter 1  Introduction 33
  Overview .................................................................... 33
    Solutions .................................................................. 34
    Capabilities .......................................................... 34
    Offline product integration ...................................... 34
  Features and functions ............................................ 36
    Fast Path Reorg/EP ................................................ 36
    Fast Path Analyzer/EP .......................................... 37
    Common features .................................................. 38
  Operational considerations ....................................... 39
    System requirements ............................................. 39
    Maintenance ......................................................... 39
  More information about Fast Path DEDBs ................ 40

Chapter 2  System resources and performance 41
  Processing modes ..................................................... 41
  Offline processing .................................................... 43
    Minimum execution JCL ......................................... 43
    Typical execution JCL ............................................ 43
    Region size ......................................................... 44
    STEPLIB requirements ......................................... 44
    Identifying the functions to be performed ................ 44
    Identifying the DEDB areas to be processed ............. 45
    Dynamic allocation of data sets ............................... 46
    Using DBRC ......................................................... 48
    Multitasking ........................................................ 50
    Caching ................................................................ 51
  Simplifying the command set .................................. 53
  Randomizer module interface ................................. 54
  Dynamically modifying messages ............................. 55
    Permanently modifying messages ......................... 56
    Temporarily modifying messages ........................... 56
<table>
<thead>
<tr>
<th>Chapter 3</th>
<th>Reorganizing a DEDB</th>
<th>61</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reorganization function overview</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>Selecting UOWs with patented intelligent reorg</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>User-specified control of the reorganization</td>
<td>63</td>
<td></td>
</tr>
<tr>
<td>Automatic pointer validation during reorganization</td>
<td>63</td>
<td></td>
</tr>
<tr>
<td>Reorganization function restrictions</td>
<td>63</td>
<td></td>
</tr>
<tr>
<td>Reorganization function inputs and outputs</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>Offline reorganization considerations</td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>Control statement</td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>Performance considerations</td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>DBRC considerations</td>
<td>66</td>
<td></td>
</tr>
<tr>
<td>Shutting down the reorganization function</td>
<td>67</td>
<td></td>
</tr>
<tr>
<td>MADS considerations</td>
<td>68</td>
<td></td>
</tr>
<tr>
<td>REORGANIZE command keywords and subcommands</td>
<td>68</td>
<td></td>
</tr>
<tr>
<td>Selecting the database and areas</td>
<td>69</td>
<td></td>
</tr>
<tr>
<td>Allocating the area data set</td>
<td>70</td>
<td></td>
</tr>
<tr>
<td>Setting UOW selection criteria</td>
<td>71</td>
<td></td>
</tr>
<tr>
<td>Selectively specifying UOWs for reorganization</td>
<td>71</td>
<td></td>
</tr>
<tr>
<td>Selecting all UOWs for reorganization</td>
<td>73</td>
<td></td>
</tr>
<tr>
<td>Controlling segment compression</td>
<td>73</td>
<td></td>
</tr>
<tr>
<td>Extending IOVF and SDEP storage portions</td>
<td>74</td>
<td></td>
</tr>
<tr>
<td>Extending IOVF</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>Extending SDEP</td>
<td>77</td>
<td></td>
</tr>
<tr>
<td>Placing segments with load control</td>
<td>78</td>
<td></td>
</tr>
<tr>
<td>Defining segments for load control placement</td>
<td>79</td>
<td></td>
</tr>
<tr>
<td>Placement control using segment content</td>
<td>81</td>
<td></td>
</tr>
<tr>
<td>Placement control using segment counts</td>
<td>82</td>
<td></td>
</tr>
<tr>
<td>Controlling error tolerance</td>
<td>83</td>
<td></td>
</tr>
<tr>
<td>Analyzing the DEDB during reorganization</td>
<td>84</td>
<td></td>
</tr>
<tr>
<td>Making an image copy during reorganization</td>
<td>85</td>
<td></td>
</tr>
<tr>
<td>Sample offline REORGANIZE command scenarios</td>
<td>85</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 4</th>
<th>Analyzing a DEDB</th>
<th>87</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis function overview</td>
<td>88</td>
<td></td>
</tr>
<tr>
<td>Offline analysis</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td>Inputs and outputs</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td>Control statement</td>
<td>91</td>
<td></td>
</tr>
<tr>
<td>DBRC considerations</td>
<td>91</td>
<td></td>
</tr>
<tr>
<td>MADS considerations</td>
<td>91</td>
<td></td>
</tr>
<tr>
<td>Analysis of an image copy</td>
<td>92</td>
<td></td>
</tr>
<tr>
<td>Inputs and outputs</td>
<td>92</td>
<td></td>
</tr>
<tr>
<td>Control statement</td>
<td>93</td>
<td></td>
</tr>
<tr>
<td>ANALYZE command keywords and subcommands</td>
<td>93</td>
<td></td>
</tr>
<tr>
<td>Selecting the Database and Areas</td>
<td>94</td>
<td></td>
</tr>
<tr>
<td>Allocating the area data set</td>
<td>95</td>
<td></td>
</tr>
</tbody>
</table>
Controlling segment expansion ........................................ 217
Controlling segment compression ................................... 218
Selecting SDEP processing method .................................. 218
Selecting database segments for input .............................. 219
  Including and excluding segments for input ...................... 219
  Narrowing the selection ........................................... 220
Changing segment contents ......................................... 223
Processing considerations for randomization of root segments to unprocessed areas .................................................. 224
Handling exceptions .................................................. 225
  Assigning an exception limit ...................................... 226
  Writing to a discarded data file .................................. 227
Rebuilding indexes during reload ................................... 229
Analyzing the DEDB during reload .................................. 229
Making an image copy during reload ............................... 230
Enhancing performance ............................................ 231
  Reload function output caching .................................. 231
  Reload function thread processing ............................... 231
  Using SORT keyword with reload ................................ 232
  Using the file sort utility ......................................... 232
  Using SORT_OPTION keyword with reload ..................... 232
UNLOAD and RELOAD program extensions ....................... 233
Sample UNLOAD and RELOAD command scenarios ............... 233

Chapter 7 Extracting data from a DEDB ............................. 235
Data extract function overview ...................................... 236
  How extract works ................................................ 236
  Format flexibility .................................................. 236
  Benefits ............................................................ 237
Data extract function inputs and outputs ......................... 237
DBRC considerations ................................................ 238
EXTRACT command keywords and subcommands .................. 239
Process overview .................................................... 240
Selecting the extract file format ................................... 241
  EXTRACT_FORMAT=EXTRACT ..................................... 242
  EXTRACT_FORMAT=HDUNLOAD .................................... 243
  EXTRACT_FORMAT=USER .......................................... 244
Selecting the database and areas ................................... 245
Allocating the input area data set .................................. 246
Allocating the extract output data set ............................. 247
  Creating a separate output data set for each area .......... 247
  Combining multiple areas into a single data set .......... 249
Selecting database segments for input and output ............... 251
  Including and excluding segments for input .................. 251
  Narrowing the selection ......................................... 252
  Outputting segment records ..................................... 255
  Samples of INCLUDE, EXCLUDE, and OUTPUT subcommands 256
Controlling segment expansion ..................................... 259
Controlling content of segment records .......................... 259
<table>
<thead>
<tr>
<th>Contents</th>
<th>11</th>
</tr>
</thead>
</table>

### Chapter 8 Initializing a DEDB

- Initialization function overview ........................................ 265
- Initialization restrictions .............................................. 266
- Initialization function inputs and outputs .......................... 266
- DBRC considerations ..................................................... 267
- INITIALIZE command keywords and subcommands .................. 268
- Selecting the database and areas ....................................... 268
- Allocating the area data sets ............................................ 269
  - Using DD statements .................................................. 269
  - Using dynamic allocation to name the area data set .......... 270
  - Using the ALLOCATE subcommand to customize dynamically defined areas 270
  - ALLOCATE subcommand sample scenarios ......................... 273
- Initializing associated indexes ....................................... 274
- Producing an image copy during initialization ..................... 274
- Enhancing performance .................................................. 274
- Sample INITIALIZE command scenarios ............................... 275

### Chapter 9 Extending a DEDB

- Extend function overview .............................................. 277
  - User-specified control of the extend .............................. 278
  - Automatic pointer validation during the extend .............. 278
- Extend function restrictions .......................................... 278
  - DEBD maintenance requirements .............................. 279
- Extend function inputs and outputs ................................ 279
- Offline extend considerations ...................................... 280
  - Control statement .................................................. 280
  - Performance considerations ...................................... 280
  - DBRC considerations ............................................... 282
  - Shutting down the Extend function ............................ 282
  - MADS considerations ............................................. 282
- EXTEND command keywords and subcommands .................... 283
- Selecting the database and areas .................................... 283
- Allocating the area data set ......................................... 285
- Specifying the execution mode ....................................... 285
- Extending IOVF and SDEP storage portions ......................... 286
  - Extending IOVF .................................................... 286
  - Extending SDEP .................................................... 288
- Extending IOVF and SDEP by using cylinders ...................... 289
- Analyzing the DEDB during an extend ............................... 290
Making an image copy during an extend ........................................ 291
Sample offline EXTEND command scenarios ............................. 291

Chapter 10 Making an image copy ................................................. 293
Offline image copy function overview ....................................... 293
Image copy function inputs and outputs .................................. 294
IC subcommand keywords and supported commands ................. 294
Selecting the database and areas .......................................... 295
  Making an image copy during analysis ................................ 295
  Making an image copy during reorganization .................... 296
  Making an image copy of a changed DEDB ....................... 296
  Making an image copy during unload ............................ 297
  Making an image copy during reload ............................ 297
Allocating the image copy data set ...................................... 298
  Using JCL to allocate the image copy data set ................ 298
  Using dynamic allocation for the image copy data set ........ 299
Stacking image copy data sets .......................................... 299
Generating multiple output image copies .............................. 301
Recording the image copy within DBRC ................................. 301
Compressing the output image copy .................................... 302
Sample IC subcommand scenarios ....................................... 303

Chapter 11 Statistics repository facility .................................. 305
Repository overview ............................................................ 306
  Repository contents ..................................................... 306
  Repository inputs and outputs ..................................... 307
Allocation rules ................................................................. 308
  Allocation rule parameters ........................................ 308
  Default allocation rule ............................................. 310
Processing the repository .................................................. 310
Creating a repository catalog ............................................ 311
Creating and modifying allocation rules ................................. 312
  Listing allocation rules ............................................ 313
  Adding allocation rules ........................................... 314
  Modifying allocation rules ...................................... 315
  Deleting allocation rules ......................................... 316
Activating repository processing ......................................... 317
  Activating the repository by using JCL .......................... 317
  Activating the repository by using dynamic allocation of the catalog on
    OPTIONS command ............................................. 318
  Activating dynamic allocation of the repository by using DFSMDA member. 318
  Disabling dynamic allocation of the repository .................... 319
  Specifying the repository group .................................. 319
Creating statistics data sets .............................................. 320
  Allocating the statistics data set ................................ 320
  Specifying the detail level ........................................ 321
  Controlling retention of repository elements ...................... 322
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Managing statistics catalog entries</td>
<td>322</td>
</tr>
<tr>
<td>Listing statistics catalog entries</td>
<td>323</td>
</tr>
<tr>
<td>Adding statistics catalog entries</td>
<td>324</td>
</tr>
<tr>
<td>Deleting statistics catalog entries</td>
<td>324</td>
</tr>
<tr>
<td>Retrieving statistics from the repository</td>
<td>327</td>
</tr>
<tr>
<td>Producing Reports</td>
<td>327</td>
</tr>
<tr>
<td>Producing a history file</td>
<td>328</td>
</tr>
<tr>
<td>Using the Fast Path Manager/EP ISPF panels</td>
<td>329</td>
</tr>
<tr>
<td>Sample repository scenarios</td>
<td>334</td>
</tr>
<tr>
<td>Appendix A Diagnostic procedures and tools</td>
<td>335</td>
</tr>
<tr>
<td>Overview</td>
<td>335</td>
</tr>
<tr>
<td>Note software product levels</td>
<td>336</td>
</tr>
<tr>
<td>Locate generated SVC dump</td>
<td>336</td>
</tr>
<tr>
<td>Define the general problem</td>
<td>338</td>
</tr>
<tr>
<td>Provide duplication and resolution information</td>
<td>339</td>
</tr>
<tr>
<td>Using diagnostic commands</td>
<td>341</td>
</tr>
<tr>
<td>Using the DUMP facility</td>
<td>341</td>
</tr>
<tr>
<td>Diagnosing problems with expressions</td>
<td>342</td>
</tr>
<tr>
<td>Appendix B Dynamic message modification</td>
<td>347</td>
</tr>
<tr>
<td>Overview</td>
<td>347</td>
</tr>
<tr>
<td>Benefits</td>
<td>348</td>
</tr>
<tr>
<td>Customization capabilities</td>
<td>348</td>
</tr>
<tr>
<td>Activation requirement</td>
<td>349</td>
</tr>
<tr>
<td>Maintenance considerations</td>
<td>349</td>
</tr>
<tr>
<td>Message modification subcommands</td>
<td>349</td>
</tr>
<tr>
<td>Dynamically modifying messages</td>
<td>350</td>
</tr>
<tr>
<td>Message severity levels</td>
<td>350</td>
</tr>
<tr>
<td>Messages eligible for dynamic modification</td>
<td>351</td>
</tr>
<tr>
<td>Changing severity level for a message</td>
<td>353</td>
</tr>
<tr>
<td>Suppressing repetitious messages</td>
<td>353</td>
</tr>
<tr>
<td>Listing message customizations</td>
<td>353</td>
</tr>
<tr>
<td>Restoring customized messages to product defaults</td>
<td>354</td>
</tr>
<tr>
<td>Activating message customizations</td>
<td>355</td>
</tr>
<tr>
<td>Activating customizations by using JCL</td>
<td>355</td>
</tr>
<tr>
<td>Activating customizations by using dynamic allocation of the catalog on</td>
<td>356</td>
</tr>
<tr>
<td>OPTIONS command</td>
<td>356</td>
</tr>
<tr>
<td>Activating dynamic allocation of customizations by using DFSMDA member</td>
<td>356</td>
</tr>
<tr>
<td>Appendix C Sample command scenarios</td>
<td>359</td>
</tr>
<tr>
<td>How to interpret the scenarios</td>
<td>360</td>
</tr>
<tr>
<td>JCL and control statement</td>
<td>361</td>
</tr>
<tr>
<td>Descriptive text</td>
<td>361</td>
</tr>
<tr>
<td>Sample library JCL</td>
<td>361</td>
</tr>
<tr>
<td>Segment hierarchy for sample DEDB</td>
<td>362</td>
</tr>
<tr>
<td>DBD for sample DEDB</td>
<td>362</td>
</tr>
</tbody>
</table>
Offline REORGANIZE command ......................................................... 363
  Simultaneously reorganize, analyze, and create image copies .............. 364
  Disable automatic analysis, place dependent segments, and stack image
  copies ......................................................................................... 365
  Specify UOW selection criteria .................................................. 366
  Extend IOVF ............................................................................. 367
  Request repository processing, reorganize areas, and store analysis
  statistics in a repository ......................................................... 368
Offline ANALYZE command .......................................................... 369
  Analyze and detect space thresholds ........................................... 370
  Analyze areas and store analysis statistics in a repository ................. 371
  Analyze areas using image copy input and create data set for area change
  modeling utility ........................................................................ 372
  Request selected reports ........................................................... 373
CHANGE command ..................................................................... 374
  Change area space and size parameters ........................................ 376
  Change DEDB compression ....................................................... 377
  Add a new segment .................................................................... 379
  Increase segment lengths and initialize fields ................................ 380
  Place dependent segments using LOADCTL ................................ 382
  Change area and add (build) associated indexes ............................ 383
  Allocate all areas for change process using existing IDCAMS area definitions 385
  Change area and customize space allocation across multiple volumes .... 386
  Control SDEP allocation on a new output area ............................... 388
UNLOAD and RELOAD commands ................................................... 390
  Use UNLOAD and RELOAD to increase area size and space parameters 392
  Use UNLOAD and RELOAD to change compression ......................... 395
  Increase number of areas and initialize new fields in segments .......... 397
  Create a test database with a subset of data .................................. 400
  Unload and reload a DEDB and add (build) associated indexes during
  reload ......................................................................................... 402
  Unload DEDB and use FABEUR6 and FABEUR7 utilities to change
  segment data, then reload DEDB ............................................... 405
  Define different space allocations for reload output areas ................ 407
Offline EXTRACT command ......................................................... 409
  Extract and output only selected segments .................................. 410
  Generate a subset of area data ................................................... 411
  Extract a selected segment and sort data ...................................... 412
  Modify segment layout during extract ........................................ 413
  Create a customized extract file format ....................................... 415
INITIALIZE command ................................................................. 418
  Initialize all areas for a database ............................................... 419
  Initialize and overwrite an existing output area ............................. 419
PROCESS_EPR command .............................................................. 421
  Create and initialize a repository catalog ..................................... 422
  Add and delete allocation rules .................................................. 423
  List statistics data sets stored in repository .................................. 424
OPTIONS command ................................................................. 425
  Request repository processing for REORGANIZE command and
  store analysis statistics in a repository .................................. 425
  Request repository processing for ANALYZE command and store analysis
  statistics in a repository ..................................................... 427
Offline EXTEND command ..................................................... 428
  Simultaneously extend IOVF, analyze, and create image copies ....... 429
  Extend IOVF and SDEP, disable automatic analysis, and stack image copies . 429
  Extend IOVF ................................................................. 430

Index 433
## Figures

- Fast Path Offline Suite System flow in offline mode ........................................... 35
- Sample JCL for multiple Fast Path Offline Suite offline functions .......................... 35
- Minimum JCL for processing of a Fast Path Offline command ............................... 43
- Typical JCL for processing of a Fast Path Offline command ............................... 43
- Specifying multiple function commands ......................................................... 45
- Identifying the DEDB area for analysis ............................................................ 45
- Control statements for separate product tasks ............................................... 50
- Control statement for serial tasks ................................................................. 51
- Control statements for parallel tasks ............................................................. 51
- Specifying the GLOBAL command .............................................................. 53
- Setting a report default ................................................................................... 54
- Specifying message overrides on the PFPOPTS DD statement .......................... 57
- Restoring selected permanent message overrides on the PFPOPTS DD statement ................................................................. 57
- Restoring all permanent message overrides on the PFPOPTS DD statement ...... 58
- Using the MESSAGE_SUPPRESSION keyword ............................................. 59
- Reorganization function inputs and outputs .................................................... 64
- Sample control statement for offline reorganization ....................................... 65
- Sample control statement for reorganizing all areas ....................................... 70
- Sample control statement for reorganizing specific areas ............................... 70
- Sample control statements for selective UOW reorganization ......................... 71
- Sample control statements for selective UOW reorganization based on CIs used . 72
- Sample control statements for selecting fragmented UOWs ......................... 72
- Sample control statements for setting minimum IOVF savings by count .......... 73
- Sample control statements for setting minimum IOVF savings by percent ......... 73
- Sample control statements for reorganizing all UOWs .................................... 73
- Using the COMPRESS keyword with the REORGANIZE command .................. 74
- Selectively specifying segments for compression with reorganize .................... 74
- Sample control statements for extending IOVF during reorganization ............. 76
- Sample control statements for extending SDEP during reorganization ............ 77
- Database hierarchical structure in LOADCTL examples .................................. 79
- Sample LOADCTL with SEGMENT only ....................................................... 80
- Sample LOADCTL for segment dependents .................................................. 80
- Sample LOADCTL for segment and its dependents ....................................... 80
- Sample LOADCTL basic selection criteria .................................................... 81
- Sample LOADCTL using qualified selection criteria ...................................... 81
- Using multiple LOADCTL subcommands to control segment placement ....... 82
- Sample LOADCTL using multiple WHERE statements for the same segment type ................................................................. 82
Sample LOADCTL using multiple WHERE statements for the same segment type .......................................................................................... 150
Sample LOADCTL using segment occurrences ................................................. 151
Sample LOADCTL using segment occurrences and qualified selection criteria .... 151
Using the EXPAND keyword with CHANGE .................................................. 151
Selectively specifying segments for expansion with CHANGE ......................... 152
Using the SDEP_PROCESS keyword with CHANGE ........................................ 153
Using the SUBSET_POINTERS keyword ....................................................... 154
Using the CHANGE command with INCLUDE and EXCLUDE subcommands .......................................................... 155
Typical DEDB segment hierarchy ................................................................. 156
CHANGE command using INCLUDE, EXCLUDE, and WHERE ....................... 157
Qualifying field references using WHERE keyword .......................................... 157
CHANGE command with SAMPLE_INTERVAL and SAMPLE_LIMIT keywords .......................................................... 158
Modifying output segment contents .............................................................. 159
Using WHERE keyword to conditionally specify segment content ................... 159
Using BYPASS_RECORD keyword to terminate CHANGE processing when a root randomizes to an area not specified on OAREA keyword .......... 161
Requesting index rebuild during DEDB restructuring ...................................... 161
Requesting index threading during DEDB restructuring ................................... 161
Requesting analysis during DEDB change ..................................................... 162
Requesting an image copy during DEDB change ........................................... 163
Unload function inputs and outputs .............................................................. 174
Sample control statement for unloading all areas ........................................... 177
Sample control statement for unloading specific areas .................................... 177
UNLOAD command with dynamic allocation for input .................................... 179
JCL to allocate an unload data set ................................................................. 180
UNLOAD command with dynamic allocation for output ................................... 180
Dynamic allocation of unload data sets ......................................................... 181
Using a generation data set group ................................................................. 182
Unloading entire area when using generalized randomizer ............................. 183
Specifying one output file when using generalized randomizer ...................... 183
Specifying multiple output files when using generalized randomizer .............. 184
Creating a separate unload file per area when using generalized randomizer .... 184
Using the EXPAND keyword with unload .................................................... 185
Selectively specifying segments for expansion with unload ......................... 185
Using the COMPRESS keyword with unload ............................................... 186
Selectively specifying segments for compression with unload ....................... 186
Using the SDEP_PROCESS keyword ........................................................... 187
Using the SUBSET_POINTERS keyword ..................................................... 188
Using the UNLOAD command with INCLUDE and EXCLUDE subcommands .......................................................... 190
Typical DEDB segment hierarchy ................................................................. 191
UNLOAD command using INCLUDE, EXCLUDE, and WHERE ....................... 191
Qualifying field references using WHERE keyword .......................................... 191
UNLOAD command with SAMPLE_INTERVAL and SAMPLE_LIMIT keywords .......................................................... 192
Modifying output segment contents .............................................................. 193
Using WHERE keyword to conditionally specify segment content ........................................... 193
Using BYPASS_RECORD keyword to terminate UNLOAD processing when a root randomizes to an area not specified on OAREA keyword ........................................... 194
Using the ERROR_THRESHOLD keyword to control error tolerance ........................................ 196
Requesting an image copy during unload processing .............................................................. 197
Reload function inputs and outputs .......................................................................................... 198
DBRC command for setting the Recovery Needed flag ............................................................ 199
Sample control statement for reloading all areas from the input unload files ......................... 202
Sample control statement for reloading specific areas from the input unload files ..................... 203
Reloading with two input data sets ............................................................................................ 205
Sample RELOAD using IOVF_LOAD_HWM ............................................................................. 211
Database hierarchical structure in LOADCTL examples ............................................................ 212
LOADCTL with SEGMENT only .................................................................................................. 213
Sample LOADCTL for segment dependents ............................................................................. 214
Sample LOADCTL for segment and its dependents ................................................................. 214
Sample LOADCTL basic selection criteria ............................................................................... 214
Sample LOADCTL using qualified selection criteria ............................................................... 215
Using multiple LOADCTL subcommands to control segment placement ............................... 215
Sample LOADCTL using multiple WHERE statements for the same segment type .................. 216
Sample LOADCTL using segment occurrences ....................................................................... 216
Sample LOADCTL using segment occurrences and qualified selection criteria ....................... 217
Using the EXPAND keyword with reload ............................................................................... 217
Selectively specifying segments for expansion with reload ....................................................... 217
Using the COMPRESS keyword with reload ........................................................................... 218
Selectively specifying segments for compression with reload .................................................. 218
Using the SDEP_PROCESS keyword with RELOAD ............................................................... 219
Using the RELOAD command with INCLUDE and EXCLUDE subcommands ......................... 220
SEGA root segment and its dependents .................................................................................... 221
RELOAD command using INCLUDE, EXCLUDE, and WHERE ............................................... 221
Qualifying field references using WHERE keyword .................................................................... 222
RELOAD command with SAMPLE_INTERVAL and SAMPLE_LIMIT keywords ......................... 222
Modifying output segment contents ........................................................................................ 224
Using WHERE keyword to conditionally specify segment content .......................................... 224
Using BYPASS_RECORD keyword to terminate RELOAD processing when a root randomizes to an area not specified on OAREA keyword ........................................... 225
Sample syntax for assigning an exception limit value .............................................................. 227
Sample syntax for writing to a discard file ................................................................................ 228
Requesting index rebuild during DEDB reload ....................................................................... 229
Requesting index threading during DEDB reload ..................................................................... 229
Control Statements for Requesting Analysis during Reload .................................................... 230
Requesting an image copy during reload .................................................................................. 231
Data extract function inputs and outputs .................................................................................. 238
Extract data control process ..................................................................................................... 240
Map of the EXTRACT format extract output file ..................................................................... 242
Map of the HDUNLOAD format extract output file ................................................. 243
Map of the USER format extract output file ......................................................... 244
Sample control statement for extracting data from all areas ................................. 245
Sample control statement for extracting data from specific areas ......................... 245
Allocating extract input using INPUT_DSN_MASK keyword ................................. 246
JCL to allocate extract data sets ............................................................................ 248
Dynamic allocation of extract data sets by using OUTPUT_DSN_MASK keyword ......... 248
JCL to allocate an extract data set by using the DDNAME keyword ......................... 249
Dynamic allocation of extract data sets by using DSNAME keyword ....................... 250
Using the EXTRACT command with INCLUDE and EXCLUDE subcommands .......... 252
EXTRACT Command using INCLUDE, EXCLUDE, and WHERE ............................... 253
EXTRACT command using INCLUDE statements that identify root keys and invoke the randomizer ................................................................. 254
Qualifying field references using WHERE keyword .............................................. 254
EXTRACT command with SAMPLE_INTERVAL and SAMPLE_LIMIT keywords .......... 255
Using the OUTPUT subcommand to control generation of segment records .......... 256
Using the WHERE keyword to conditionally specify segment output ................. 256
SEGA Root Segment and its Dependents ............................................................... 257
Selectively specifying segments for expansion with extract .................................. 259
EXTRACT command with OUTPUT subcommand and FIELDS keyword .............. 260
Using the WHERE keyword to conditionally specify segment content ............... 261
Using SEGMENT_RECORD_PREFIX keyword to control content of segment record prefix ................................................................. 261
Using SEGMENT_RECORD_SUFFIX keyword to control content of segment record suffix ................................................................. 262
Using BREAK and FIELDS keywords to create database header and trailer records ................................................................. 263
Initialization Function Inputs and Outputs ............................................................. 266
DBRC command for setting the Recovery Needed flag on .................................... 267
Sample control statement for initializing all areas ............................................. 269
Sample control statement for initializing specific areas ..................................... 269
Requesting index initialization ............................................................................. 274
Requesting an image copy during analysis .......................................................... 274
Extend function inputs and outputs ...................................................................... 279
Sample control statement for an offline extend of IOVF .................................... 280
Sample control statement for extending IOVF for all areas ......................... 284
Sample control statement for extending SDEPs for specific areas .................. 284
Sample control statements for extending IOVF ................................................. 287
Sample control statements for extending SDEP .............................................. 289
Sample control statements for extending IOVF by 100 cylinders and SDEP by 50 cylinders ................................................................. 289
Control statements for requesting analysis during an extend ............................ 291
Requesting an image copy during an extend ...................................................... 291
Image copy function inputs and outputs ............................................................. 294
Specifying the image copy function with the ANALYZE command ................ 296
Specifying the image copy function with the REORGANIZE command ............. 296
Specifying the image copy function with the CHANGE command ........................................ 297
Specifying the image copy function with the UNLOAD command ................................. 297
Specifying the image copy function with the RELOAD command ................................. 297
JCL to allocate an image copy data set ............................................................................. 298
Dynamic allocation of an image copy data set .................................................................. 299
Using a generation data set group .................................................................................... 299
Using the STACK_NAME keyword ................................................................................... 300
Illegal use of the same stack name within a single command ........................................ 300
Effect of stacked image copies on parallel processing ..................................................... 301
Specifying multiple image copy data sets for each area ................................................... 301
Recording image copies with DBRC ................................................................................ 302
Compressing the image copy data set .............................................................................. 303
Repository Processing ....................................................................................................... 307
Listing selected allocation rules ....................................................................................... 313
Adding a group allocation rule for a repository ............................................................... 314
Modifying parameters for selected allocation rules ......................................................... 316
Deleting an area allocation rule from a repository ........................................................... 316
Activating the repository by using JCL ............................................................................. 317
Activating the repository by using REPOSITORY_DSNAME on OPTIONS command .......... 318
Activating the repository by using DFSMDA member ..................................................... 318
Sample JCL for specifying the REPOSITORY_OVERWRITE keyword ......................... 321
Sample command for writing statistics information to a data set .................................. 324
Adding a statistics catalog entry to a repository ............................................................. 324
Specifying the REPOSITORY_RETENTION_PERIOD keyword under the PFPOPTS DD statement ........................................................................................................ 325
Specifying the REPOSITORY_RETENTION_PERIOD keyword and the REPOSITORY_RETENTION_COUNT keyword under the PFPOPTS DD statement .......................... 326
Automatic deletion of entries using PFPEPR00 ............................................................ 326
Retrieving statistics using PFPEPR00 program .............................................................. 328
Producing history data using PFPEPR00 program ......................................................... 328
Producing history data using PFPEPR00 program ......................................................... 328
FAST PATH MANAGER/EP main menu ........................................................................ 330
FAST PATH ANALYZER/EP options menu .................................................................... 330
Report Options panel ....................................................................................................... 331
Statistics Deletion Confirmation Panel ........................................................................... 332
Report List panel ............................................................................................................... 333
Sample symptom dump .................................................................................................... 339
Problem Summary Form .................................................................................................. 340
Problem Documentation Form ....................................................................................... 341
Example exception report for common data exceptions ............................................... 343
Example of a complete exception report ........................................................................ 344
Changing severity level for specific message ............................................................... 353
Specifying a suppression threshold for specific message ............................................... 353
Listing all message customizations stored in repository catalog ................................... 354
Restoring all message customizations to product defaults ............................................. 354
Restoring selected message customizations to product defaults .................................... 354
Activating message customizations by using JCL ............................................................ 355
Activating message customizations by using REPOSITORY_DSNAME with OPTIONS command .......................................................... 356
Activating the Repository by Using DFSMDA Member ......................................................... 356
JCL and control statement .................................................................................................... 361
Segment hierarchy for sample DEDB ................................................................................... 362
DBD for sample DEDB ........................................................................................................ 362
JCL to simultaneously reorganize, analyze, and create image copies ................................ 364
JCL to disable automatic analysis, place dependent segments, and stack image copies ...... 365
JCL to specify UOW selection criteria .................................................................................. 366
JCL to extend IOVF ............................................................................................................. 367
JCL to request repository processing, reorganize areas, and store analysis statistics in a repository ........................................................................................................................................ 368
JCL to analyze and detect space thresholds ........................................................................ 370
JCL to analyze areas and store analysis statistics in a repository ........................................ 371
JCL to analyze areas using image copy input and create data set for area change modeling utility ........................................................................................................................... 372
JCL to request selected reports ........................................................................................... 373
JCL to change area space and size parameters ................................................................. 376
JCL to change DEDB compression ...................................................................................... 378
JCL to add a new segment .................................................................................................. 379
JCL to increase segment lengths and initialize fields ......................................................... 380
JCL to place dependent segments using LOADCTL ......................................................... 382
JCL to change area and add (build) associated indexes .................................................... 383
JCL to allocate all areas for change process using existing IDCAMS area definitions .......... 385
JCL to change area and customize space allocation across multiple volumes ................. 386
DEDB area—RAA portion ................................................................................................. 388
JCL to control SDEP allocation on a new output area ...................................................... 388
DEDB area—VOLCNT ........................................................................................................ 390
JCL to use UNLOAD and RELOAD to increase area size and space parameters .................. 392
JCL to use UNLOAD and RELOAD to change compression ............................................. 395
JCL to increase numbers of areas and initialize new fields in segments ......................... 398
JCL to create a test database with a subset of data ............................................................ 400
JCL to unload and reload a DEDB and add (build) associated indexes during reload .......... 402
JCL to unload DEDB and use FABEUR6 and FABEUR7 utilities to change segment data, then reload DEDB ................................................................. 405
JCL to define different space allocations for reload output areas ...................................... 408
JCL to extract and output only selected segments ............................................................ 410
JCL to generate a subset of area data .................................................................................. 411
JCL to extract a selected segment and sort data ............................................................... 412
JCL to modify segment layout during extract .................................................................... 413
JCL to create a customized extract file format ................................................................. 415
JCL to initialize all areas for a database ............................................................................. 419
JCL to initialize and overwrite an existing output area ..................................................... 419
DEDB area—division of PFPSAMP1 ................................................................................. 421
JCL to create and initialize a repository catalog ............................................................... 422
| JCL to add and delete allocation rules | 423 |
| JCL to list statistics data sets stored in repository | 424 |
| JCL to request repository processing for REORGANIZE command and store analysis statistics in a repository | 425 |
| JCL to request repository processing for ANALYZE command and store analysis statistics in a repository | 427 |
| JCL to simultaneously extend IOVF, analyze, and create image copies | 429 |
| JCL to extend IOVF and SDEP, disable automatic analysis, and stack image copies | 429 |
| JCL to extend IOVF | 430 |
Tables

Synergy between Fast Path Offline products and other BMC products for IMS ........................................... 35
Available operational modes by Fast Path/EP command ................................................................. 42
Area data set disposition for Fast Path Offline Suite command processes ........................................ 48
PFPOPTS OVERRIDE subcommand keywords .................................................................................. 56
REORGANIZE command keywords and subcommands ................................................................ 68
ANALYZE command keywords and subcommands ......................................................................... 93
Fast Path Analyzer/EP characteristics, report titles, and keywords .................................................. 104
Threshold types and keywords ........................................................................................................ 109
DBRC settings for change processing ............................................................................................... 133
CHANGE command keywords and subcommands ......................................................................... 134
Using INCLUDE, EXCLUDE and OUTPUT subcommands with the CHANGE command ...................... 159
DBRC settings for the unload function ............................................................................................. 175
UNLOAD command keywords and subcommands ......................................................................... 175
Using INCLUDE, EXCLUDE and OUTPUT subcommands with the UNLOAD Command ......................... 195
DBRC settings and authorization levels for reload processing ........................................................... 199
RELOAD command keywords and subcommands ......................................................................... 200
DBRC settings for the extract function ............................................................................................. 238
EXTRACT command subcommands and keywords ....................................................................... 239
Data extract tailoring options according to value specified for EXTRACT_FORMAT keyword .............. 241
Using INCLUDE, EXCLUDE and OUTPUT subcommands with the EXTRACT command and EXTRACT_FORMAT=EXTRACT or EXTRACT_FORMAT=USER ......................................................... 257
Using INCLUDE, EXCLUDE and OUTPUT subcommands with the EXTRACT command and EXTRACT_FORMAT=HDUNLOAD .................................................................................................................. 258
Available options for BREAK keyword .......................................................................................... 262
DBRC setting for the initialization function ........................................................................................ 267
INITIALIZE command keywords .................................................................................................... 268
EXTEND command keywords and subcommands ........................................................................ 283
Command keywords for the IC subcommand .................................................................................. 295
Repository statistics data set name mask substitution parameters ..................................................... 308
Allocation commands and subcommands available with PFPEPR00 .............................................. 312
Type parameter values .................................................................................................................... 313
Available keywords for the LIST subcommand ............................................................................. 313
Available keywords for the ADD subcommand ............................................................................. 314
Type parameter values .................................................................................................................... 315
Available keywords for the MODIFY subcommand ....................................................................... 315
Available keywords for the DELETE subcommand ........................................................................ 316
Statistics management and retrieval commands and subcommands available with PFPEPR00 .................................................. 322
Available keywords for the LIST subcommand ............................................. 323
Available keywords for the RETRIEVE subcommand .................................... 327
Reason codes for RC=8 on message BMC110980E ....................................... 336
DUMP command set .................................................................................. 342
Expression exception codes ....................................................................... 343
Message modification commands and subcommands available with PFPEPR00 . 350
Available keywords for the OVERRIDE subcommand .................................... 350
Fast Path Offline Suite message severity levels ............................................ 351
Fast Path Offline Suite messages available for dynamic modification ............ 351
Descriptive text for JCL ............................................................................. 354
REORGANIZE scenarios ........................................................................... 363
Descriptive text for JCL to simultaneously reorganize, analyze, and create
image copy ............................................................................................... 365
Descriptive text for JCL to disable automatic analysis, place dependent
segments, and stack image copies ............................................................... 366
Descriptive text for JCL to specify UOW selection criteria ............................. 367
Descriptive text for JCL to extend IOVF ...................................................... 368
Descriptive text for JCL to request repository processing, reorganize areas,
and store analysis statistics in a repository ............................................... 369
ANALYZE scenarios ................................................................................ 370
Descriptive text for JCL to analyze and detect space thresholds .................... 371
Descriptive text for JCL to analyze areas and store analysis statistics in a
repository ................................................................................................. 372
Descriptive text for JCL to analyze areas using image copy input and create
data set for area change modeling utility .................................................. 373
Descriptive text for JCL to request selected reports ...................................... 374
CHANGE scenarios .................................................................................. 374
Descriptive text for JCL to change area space and size parameters ................. 377
Descriptive text for JCL to change DEDB compression ................................ 378
Descriptive text for JCL to add a new segment ............................................ 380
Descriptive text for JCL to increase segment lengths and initialize fields ....... 381
Descriptive text for JCL to place dependent segments using LOADCTL ....... 382
Descriptive text for JCL to change area and add (build) associated indexes ...... 384
Descriptive text for JCL to allocate all areas for change process using existing
IDCAMS area definitions ......................................................................... 386
Descriptive text for JCL to change area and customize space allocation across
multiple volumes ....................................................................................... 387
Descriptive text for JCL to control SDEP allocation on a new output area ........ 389
UNLOAD and RELOAD scenarios ............................................................... 391
Descriptive text for JCL to use UNLOAD and RELOAD to increase area size
and space parameters .............................................................................. 393
Descriptive text for JCL to use UNLOAD and RELOAD to change compression in
segments .................................................................................................. 396
Descriptive text for JCL to increase numbers of areas and initialize new fields
in segments ............................................................................................... 399
Descriptive text for JCL to create a test database with a subset of data .......... 401
Descriptive text for JCL to unload and reload a DEDB and add (build)
associated indexes during reload ............................................................... 403
Descriptive text for JCL to unload DEDB and use FABEUR6 and FABEUR7 utilities to change segment data, then reload DEDB .................................................. 406
Descriptive text for JCL to define different space allocations for reload output areas ................................................................. 408
EXTRACT scenarios ........................................................................ 409
Descriptive text for JCL to extract and output only selected segments ........ 410
Descriptive text for JCL to generate a subset of area data ....................... 412
Descriptive text for JCL to extract a selected segment and sort data .......... 413
Descriptive text for JCL to modify segment layout during extract ............... 414
Descriptive text for JCL to create a customized extract file format ............ 416
Data output from JCL to create a customized extract file format ................. 417
INITIALIZE scenarios .................................................................. 418
Descriptive text for JCL to initialize all areas for a database ..................... 419
Descriptive text for JCL to initialize and overwrite an existing output area ...... 420
PROCESS_EPR scenarios ............................................................... 422
Descriptive text for JCL to create and initialize a repository catalog ............ 423
Descriptive text for JCL to add and delete allocation rules ....................... 424
Descriptive text for JCL to list statistics data sets stored in repository .......... 424
OPTIONS scenarios ...................................................................... 425
Descriptive text for JCL to request repository processing for REORGANIZE command and store analysis statistics in a repository ............... 426
Descriptive text for JCL to request repository processing for ANALYZE command and store analysis statistics in a repository ...................... 427
EXTEND scenarios ...................................................................... 428
Descriptive text for JCL to simultaneously extend IOVF, analyze, and create image copies ................................................................. 429
Descriptive text for JCL to extend IOVF and SDEP, disable automatic analysis, and stack image copies ...................................................... 430
Descriptive text for JCL to extend IOVF ............................................ 431
The Fast Path Offline Suite of products provides tools for the management, maintenance, and performance tuning of data entry databases (DEDBs) for the IBM® IMS™ environment. These tools deliver a wide range of functions for completing database reorganizations and managing space usage offline during a batch maintenance window. The Fast Path Offline Suite consists of the following products:

- Fast Path Reorg/EP
- Fast Path Analyzer/EP

This book contains detailed information about the Fast Path Offline Suite of products and is intended for IMS database administrators and technical support personnel. This book provides detailed procedures for executing the major functions that are provided by these two products. It also describes how to combine elements of the Fast Path/EP command language to customize these functions to meet your needs.

Like most BMC documentation, this book is available in printed and online formats. To request printed books or to view online books and notices (such as release notes and technical bulletins), see the support website at http://www.bmc.com/support.

---

**NOTE**

Online books are formatted as PDF or HTML files. To view, print, or copy PDF books, use the free Adobe Reader from Adobe Systems. If your product installation does not install the reader, you can obtain the reader at http://www.adobe.com.

The software also offers online Help. To access Help, press F1 within any product or click the Help button in graphical user interfaces (GUIs).
# Related publications

The following related publications supplement this book:

<table>
<thead>
<tr>
<th>Document</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC Products for IMS Installation Guide</td>
<td>provides installation procedures and optional post-installation procedures for database administrators and technical support personnel involved with the initial installation and maintenance installation of the Fast Path Offline Suite and other BMC Fast Path products for IMS</td>
</tr>
<tr>
<td>Database Products for IMS Configuration Guide</td>
<td>provides optional post-installation procedures for implementing BMC Fast Path products for IMS after installation</td>
</tr>
<tr>
<td>Fast Path/EP Series Messages Manual</td>
<td>provides descriptions of all warning messages and error messages that can be generated by the Fast Path Offline Suite and other BMC Fast Path products for IMS</td>
</tr>
<tr>
<td>Fast Path/EP Series Reference Manual</td>
<td>provides reference information on the following topics:</td>
</tr>
<tr>
<td></td>
<td>■ JCL conventions</td>
</tr>
<tr>
<td></td>
<td>■ conventions for using the individual commands, subcommands, and keywords that control product functionality</td>
</tr>
<tr>
<td></td>
<td>■ syntax rules for coding scripts and expressions that are used in command language</td>
</tr>
<tr>
<td></td>
<td>■ field descriptions of the reports that you can generate by using the Fast Path Analyzer/EP product</td>
</tr>
<tr>
<td></td>
<td>■ use of supporting utilities and program extensions</td>
</tr>
</tbody>
</table>

From the BMC Support Central website ([http://www.bmc.com/support](http://www.bmc.com/support)), you can

- download a zipped set of documentation PDFs from each product’s EPD page
- link to the BMC Documentation Center ([https://webapps.bmc.com/infocenter/index.jsp](https://webapps.bmc.com/infocenter/index.jsp)) to browse documentation sets, or to view video demos (short overviews of selected product concepts, tasks, or features)
- view individual product documents (books and notices) within the “A – Z Supported Product List”

You can order hardcopy documentation from your BMC sales representative or from the support site. You can also subscribe to proactive alerts to receive e-mail messages when notices are issued.
Conventions

This book uses the following special conventions:

- All syntax, operating system terms, and literal examples are presented in this typeface.

- Variable text in path names, system messages, or syntax is displayed in *italic* text:

  testsys/instance/fileName

- The symbol => connects items in a menu sequence. For example, **Actions => Create Test** instructs you to choose the **Create Test** command from the **Actions** menu.

- Revision bars in the document mark changes that clarify or correct existing information or that provide new information. Revision bars do not mark editorial changes, formatting changes, or corrections of typographical errors unless these updates significantly affect your use of the information.

Syntax statements

The following example shows a sample syntax statement:

```
COMMAND KEYWORD1 [KEYWORD2 | KEYWORD3] KEYWORD4={YES | NO} fileName...
```

The following table explains conventions for syntax statements and provides examples:

<table>
<thead>
<tr>
<th>Item</th>
<th>Example</th>
</tr>
</thead>
</table>
| Items in *italic* type represent variables that you must replace with a name or value. If a variable is represented by two or more words, initial capitals distinguish the second and subsequent words. | alias  
databaseDirectory  
serverHostName                                                                 |
| Brackets indicate a group of optional items. Do not type the brackets when you enter the option. A comma means that you can choose one or more of the listed options. You must use a comma to separate the options if you choose more than one option. | [tableName, columnName, field]  
[-full, -incremental, -level] (UNIX) |
Summary of changes

For detailed information about enhancements, changes, and corrections that are included in your version of the product, see the product release notes. The release notes are available from the BMC Support Central page [http://www.bmc.com/support](http://www.bmc.com/support).

<table>
<thead>
<tr>
<th>Item</th>
<th>Example</th>
</tr>
</thead>
</table>
| Braces indicate that at least one of the enclosed items is required. Do not type the braces when you enter the item. | {DBDName | tableName}  
UNLOAD device={disk | tape,  
fileName | deviceName}  
{-a | -c} (UNIX) |
| A vertical bar means that you can choose only one of the listed items. In the example, you would choose either commit or cancel. | {commit | cancel}  
{-commit | -cancel} (UNIX) |
| An ellipsis indicates that you can repeat the previous item or items as many times as necessary. | columnName . . . |
Introduction

The Fast Path Offline Suite of products provides offline maintenance solutions for IMS data entry databases (DEDBs).

This chapter discusses the following topics:

Overview ................................................................. 33
Solutions ............................................................... 34
Capabilities ......................................................... 34
Offline product integration .................................. 34
Features and functions ......................................... 36
  Fast Path Reorg/EP .................................................. 36
  Fast Path Analyzer/EP .......................................... 37
  Common features ................................................. 38
Operational considerations .................................. 39
  System requirements .......................................... 39
  Maintenance ....................................................... 39
More information about Fast Path DEDBs ............... 40

Overview

Applications with high transaction rates demand fast transaction throughput and are quite often the same applications with voluminous storage requirements. Many of the world’s most business-critical applications use IMS Fast Path to take advantages of the transaction rate, large storage capabilities, and speed characteristics of DEDBs.

For customers who are accustomed to maintaining DEDBs during a batch maintenance window, BMC has created the Fast Path Offline Suite to provide more efficient and effective solutions for DEDB analysis, performance, space management, and capacity management. These products help minimize downtime required for offline maintenance processes. The Fast Path Offline Suite also lets you perform analysis and maintenance tasks simultaneously.
**Solutions**

The Fast Path Offline Suite consists of Fast Path Analyzer/EP and Fast Path Reorg/EP. While these products are available as stand-alone products, they provide significant advantages when applied as a comprehensive offline DEDB maintenance suite.

**Capabilities**

The Fast Path Offline Suite offers the following capabilities:

- eliminate manual steps for completing database reorganizations
- perform “intelligent” database reorganizations (reorganize only the portions of the database that actually need reorganizing)
- eliminate time-consuming unloads and reloads traditionally required for DEDB space reclaim
- eliminate time-consuming unloads and reloads that are traditionally required for DEDB structure change
- analyze the content and structure of DEDBs
- significantly reduce I/O through concurrent reorganization, analysis, and image-copying of a DEDB

**Offline product integration**

This section discusses the advantages of combining and using the two products that comprise the Fast Path Offline Suite in an environment in which maintenance is handled within the constraints of the traditional batch processing window.

*Figure 1* and *Figure 2* show the functional flow and simple one-step JCL that is used for simultaneous execution of Fast Path Offline Suite database reorganization, analysis, and image copy functions in offline mode.
Figure 1  Fast Path Offline Suite System flow in offline mode

![Fast Path Offline Suite System flow in offline mode](image)

Figure 2  Sample JCL for multiple Fast Path Offline Suite offline functions

```
//PFP EXEC PGM=PFPMAIN,REGION=0M
//STEPLIB DD DSN=BMC.PFP.LOAD,DISP=SHR
//      DD DSN=IMS.RESLIB,DISP=SHR
//PFPSYSIN DD *
GLOBAL  POINTER_VALIDATION=QUICK
IC DSNAME='dataset-name-mask', UNIT=TAPE,DISP=(CATLG)
REORGANIZE DBD=dedb1,IAREA=ALL
REORGANIZE DBD=dedb2,IAREA=ALL
REORGANIZE DBD=dedbn,IAREA=ALL
/*
```

Table 1 provides a summary of how the products in the Fast Path Offline Suite can be used with each other and with other BMC products for IMS to save time and resources when performing Fast Path database maintenance, analysis, and recovery tasks.

Table 1  Synergy between Fast Path Offline products and other BMC products for IMS (part 1 of 2)

<table>
<thead>
<tr>
<th>BMC IMS Fast Path product</th>
<th>Related BMC IMS product</th>
<th>Benefit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast Path Reorg/EP</td>
<td>Fast Path Analyzer/EP</td>
<td>You can gather statistical information about your Fast Path database area and validate its pointers at the same time you are implementing a database reorganization, change or reload operation.</td>
</tr>
<tr>
<td>Fast Path Reorg/EP</td>
<td>DATA PACKER/IMS</td>
<td>You can dynamically expand and compress segment data during an offline reorganization.</td>
</tr>
</tbody>
</table>
Fast Path Offline Suite products let you complete a broad range of database maintenance and analysis tasks. This section provides additional detail on the features and benefits of the individual products that comprise the Fast Path Offline Suite.

**Fast Path Reorg/EP**

If you elect to perform database reorganizations within the batch maintenance window, the Fast Path Reorg/EP product provides a variety of reorganization functions to reclaim space and restructure databases with minimal database downtime. By delivering increased speed and flexibility, Fast Path Reorg/EP helps you plan and execute DEDB maintenance operations more effectively.

Key product functions are summarized as follows:

- The patented Intelligent Reorg feature recognizes and reorganizes only UOWs that require reorganization. This technique eliminates unnecessary I/O by ignoring UOWs that would not benefit from a reorganization. It also lets you fine-tune DEDB reorganization by selecting only certain UOWs for reorganization. You can select either of the following:
  - only UOWs that extend into independent overflow (IOVF)
  - only UOWs that exceed a threshold percent of disorganization
The reorganization function lets you optimize performance and reclaim space by processing only UOWs that are disorganized, leaving organized UOWs unaffected. Space reclamation is performed efficiently, without the high-I/O of unload and reload processing.

The database change function lets you rapidly alter a database, changing the DBD, without I/O-intensive unload and reload of the DEDB. The function reads the DEDB from DASD or from an image copy data set, and writes the restructured DEDB directly to DASD.

The patented Extend function lets you increase the size of IOVF and SDEP storage portions of a DEDB during reorganization without unload and reload processing.

The unload and reload function provides the capability to execute traditional offline DEDB unloads and reloads. When their use is appropriate, the unload and reload functions provide advanced features and controls as compared to traditional unload and reload processing.

The reorganize, change, unload, and reload functions let you take one or more image copies of the DEDB concurrently.

**Fast Path Analyzer/EP**

For customers who elect to perform database analysis within the batch maintenance window, the Fast Path Analyzer/EP product provides a variety of functions to analyze and produce statistics on DEDBs with minimal downtime. Key product functions are summarized as follows:

- The validation function examines, verifies and reports on pointer data, including segment pointers, root addressable pointers (RAPs), and SDEP pointers.

- The analysis function computes and analyzes information about single or multiple areas. User-specified thresholds can be used to set statistical analysis limits. An area data set or an image copy can be used as input to an analyze function.

- The analysis function lets you take one or more image copies of the DEDB concurrently.

- A variety of reports provide comprehensive analysis data to help you monitor and manage performance characteristics, space usage and physical attributes of DEDBs.

An area change modeling utility lets you model the effect of a potential DEDB change before performing it.

Auxiliary utilities that are provided with Fast Path Analyzer/EP allow VSAM zap/dumps, reporting on SDEP space usage, and invocation of DEDB randomizing routines.


Common features

Several important features that are common to all Fast Path Offline Suite products facilitate product implementation and use:

- For customers who receive the Fast Path Offline Suite products on the I-series cartridge tape, installation is accomplished with the BMC Installation System. For more information, see the BMC Products for IMS Installation Guide.

- Fast Path Offline Suite customers have the option of using the DBA Toolkit, which is included with the MAXM Database Advisor for IMS product on the I-series cartridge tape. Installation is accomplished with the BMC Installation System. For more information, see the BMC Products for IMS Installation Guide.

- Fast Path Offline Suite products incorporate an easy-to-use command language. Commands, subcommands, keywords, and keyword parameter options provide control and customization of DEDB maintenance and analysis functions. Command sets are composed of a command, optional subcommands, and optional keywords with their parameter values.

- Fast Path Offline Suite products are integrated with BMC products for IMS (IMAGE COPY PLUS, RECOVERY PLUS, RECOVERY MANAGER, DELTA/IMS, and DATA PACKER/IMS).
Operational considerations

Fast Path Offline Suite products encompass requirements and compliance.

System requirements

- IBM z/Architecture® mainframes
  
  BMC licenses each product in the Fast Path Offline Suite to run on specific CPUs.
- IBM z/OS 1.10 or later
- supported releases of IMS
- ISPF 3.3 or later
- APF authorization for the STEPLIB containing the Fast Path Offline Suite load library
- standard IMS RESLIB (SDFSRESL) for execution

Maintenance

Fast Path Offline Suite products integrate with each other and with other BMC Software IMS products:

- Fast Path/EP Series products
- DATA PACKER/IMS
- DELTA/IMS
- IMAGE COPY PLUS
- RECOVERY PLUS
- RECOVERY MANAGER

NOTE

This book lists product versions for informational purposes only. For information about supported versions of BMC products, see the BMC Software Web site at www.bmc.com.
More information about Fast Path DEDBs

For more information about Fast Path DEDBs, see the IBM Redbooks® IMS FAST PATH SOLUTIONS GUIDE (reference number: SG24-4301-00).
System resources and performance

This chapter discusses considerations that are related to the use of system resources and execution of Fast Path Offline Suite products. These considerations are prerequisite to optimizing product performance. This chapter discusses the following topics:

- Processing modes .................................................. 41
- Offline processing ................................................. 43
  - Minimum execution JCL ........................................ 43
  - Typical execution JCL ........................................... 43
  - Region size ......................................................... 44
  - STEPLIB requirements .......................................... 44
  - Identifying the functions to be performed ...................... 44
  - Identifying the DEDB areas to be processed .................... 45
  - Dynamic allocation of data sets ................................ 46
  - Using DBRC ..................................................... 48
  - Multitasking ..................................................... 50
  - Caching ........................................................... 51
  - Simplifying the command set .................................... 53
  - Randomizer module interface ................................... 54
  - Dynamically modifying messages ................................. 55
    - Permanently modifying messages ............................. 56
    - Temporarily modifying messages ............................ 56
    - Temporarily restoring permanent message customizations . 57
    - Suppressing repetitious messages by suffix type .......... 58

Processing modes

Each key command-driven function that is provided by Fast Path Offline Suite products can operate in the processing modes that are listed in Table 2. This table also lists product or products that can execute the command in the indicated mode. Because this book deals only with offline processes, the offline mode is the only mode that is discussed in this chapter.
NOTE
Executing any of the command-driven functions in online mode requires a license for a Fast Path Online Suite product. These online processes are discussed in the *Fast Path Online Suite User Guide*.

### Table 2  Available operational modes by Fast Path/EP command

<table>
<thead>
<tr>
<th>Command</th>
<th>Offline</th>
<th>Online</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANALYZE</td>
<td>X</td>
<td>X</td>
<td>Fast Path Online Analyzer/EP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Fast Path Analyzer/EP</td>
</tr>
<tr>
<td>CHANGE</td>
<td>X</td>
<td></td>
<td>Fast Path Online Reorg/EP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Fast Path Reorg/EP</td>
</tr>
<tr>
<td>DMAC_PRINT</td>
<td>X</td>
<td>X</td>
<td>utility provided with any Fast Path/EP product</td>
</tr>
<tr>
<td>EXTEND</td>
<td>X</td>
<td>X</td>
<td>Fast Path Online Reorg/EP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Fast Path Reorg/EP</td>
</tr>
<tr>
<td>EXTRACT</td>
<td>X</td>
<td>X</td>
<td>Fast Path Online Analyzer/EP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Fast Path Analyzer/EP</td>
</tr>
<tr>
<td>IMAGECOPY IC (subcommand)</td>
<td>X</td>
<td>X</td>
<td>Fast Path Online Image Copy/EP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Fast Path Reorg/EP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Fast Path Analyzer/EP</td>
</tr>
<tr>
<td>INITIALIZE</td>
<td>X</td>
<td></td>
<td>Fast Path Online Reorg/EP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Fast Path Reorg/EP</td>
</tr>
<tr>
<td>PFPSORT</td>
<td>X</td>
<td></td>
<td>Fast Path Online Reorg/EP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Fast Path Reorg/EP</td>
</tr>
<tr>
<td>PROCESS_AREA</td>
<td>X</td>
<td></td>
<td>utility provided with any Fast Path/EP product</td>
</tr>
<tr>
<td>RELOAD</td>
<td>X</td>
<td></td>
<td>Fast Path Online Reorg/EP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Fast Path Reorg/EP</td>
</tr>
<tr>
<td>REORGANIZE</td>
<td>X</td>
<td>X</td>
<td>Fast Path Online Reorg/EP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Fast Path Reorg/EP</td>
</tr>
<tr>
<td>RETRIEVE</td>
<td>X</td>
<td></td>
<td>Fast Path Online Analyzer/EP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Fast Path Analyzer/EP</td>
</tr>
<tr>
<td>UNLOAD</td>
<td>X</td>
<td></td>
<td>Fast Path Online Reorg/EP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Fast Path Reorg/EP</td>
</tr>
</tbody>
</table>

System resource and performance considerations for the execution of Fast Path Offline Suite product functions are dependent on the processing mode, the characteristics of the area being processed, and the optional processing that is requested by using command and subcommand keywords.
Offline processing

Offline mode is characterized by processing DEDB areas that are offline to all IMS control regions. In offline mode, no other utility or process can be executed concurrently by using the database areas being processed by the Fast Path Offline Suite product function.

Minimum execution JCL

Figure 3 shows the minimum JCL needed to execute Fast Path Offline Suite product functions. Dynamic allocation of area data sets, the ACB library, and RECON data sets is assumed.

```
/* Figure 3  Minimum JCL for processing of a Fast Path Offline command */
//PFP EXEC PGM=PFPMAIN,REGION=OM
//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
  // DD DISP=SHR,DSN=IMSVS.RESLIB
//PFPsysin DD *
--- control statements ---
/*
```

Typical execution JCL

Figure 4 shows typical JCL that is used to execute the Fast Path Offline Suite product functions.

```
/* Figure 4  Typical JCL for processing of a Fast Path Offline command (part 1 of 2) */
//PFP EXEC PGM=PFPMAIN,REGION=OM
//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD,DISP=SHR
  // DD DISP=SHR,DSN=IMSVS.RESLIB,DISP=SHR
  //************************************************************
  //IMSACB DD DSN=IMSVS.ACBLIB,DISP=SHR
  //*/ or
  //MODSTAT DD DSN=IMSVS.MODSTAT,DISP=SHR
  //*/ or
  //OLCSTAT DD DSN=IMSVS.OLCSTAT,DISP=SHR
  //************************************************************
  //IMSACBA DD DSN=IMSVS.ACBLIBA,DISP=SHR
  //IMSACBB DD DSN=IMSVS.ACBLIBB,DISP=SHR
  //RECON1 DD DSN=IMSVS.RECON1,DISP=SHR
  //RECON2 DD DSN=IMSVS.RECON2,DISP=SHR
  //RECON3 DD DSN=IMSVS.RECON3,DISP=SHR
```
To ensure optimum performance, BMC recommends that you request the maximum available storage for execution of Fast Path Offline Suite product functions. Maximum storage is normally requested by specifying REGION=0M on the EXEC statement for the JOB step. Some sites might have different requirements. To determine how to request the maximum REGION size, check with your z/OS systems programmer.

STEPLIB requirements

Fast Path Offline Suite product functions must execute as an APF–authorized program. The product installation library (named BMC.PFP.LOAD in all examples) must be identified as an APF–authorized library. All data sets that are listed within the STEPLIB concatenation must also be APF authorized.

BMC recommends that the product installation library be placed first within the STEPLIB concatenation. If other libraries precede the product installation library, none of these libraries can contain a member name duplicating a member within the product library, to ensure proper execution. The exception is member DFSRRC00 (found within the PFP product library), which is not used in offline mode. For more information, see the BMC Products for IMS Installation Guide and the Database Products for IMS Configuration Guide.

Identifying the functions to be performed

The PFPSYSIN control statements identify the functions to be performed. As shown in Figure 5, the DBD keyword is required for each offline function command. The DBD keyword identifies the DEDB (DBD name) to be processed by the command. Multiple DEDBs can be processed within a single job step.

--- control statements ---
/*
Identifying the DEDB areas to be processed

The IAREA keyword (OAREA for the CHANGE and RELOAD commands) identifies the names of the areas that the function will process. An example is shown in Figure 6 for the ANALYZE command. If you omit the IAREA (or OAREA) keyword, the function processes all areas that are defined as part of the database.

Areas also can be specified on the IAREA keyword by using any combination of area names, area numbers, or area ranges. The following parameters are available for the IAREA keyword:

- **IAREA=ALL** (default) or **IAREA=*** specifies all areas of the DEDB.

- **IAREA=areaname** specifies one or more areas by using the one-character to eight-character area name for each area specified. Multiple area names must be enclosed in parentheses and separated by commas.

- **IAREA=areanumber** specifies one or more areas by using the one-character to five-character area number for each area specified. Multiple area numbers must be enclosed in parentheses and separated by commas.

- **IAREA=(RANGE=(startarea,endarea))** specifies a consecutive range of areas using either areaname or areanumber parameters. The area number associated with startarea must be less than the area number associated with endarea.

An asterisk (*) can be used to specify all areas of the DEDB. When the * character is used with the RANGE keyword, it can be used to specify the beginning or ending range for specific areas of the DEDB.

Detailed examples of complex IAREA statements are provided in other chapters in this book.
**Dynamic allocation of data sets**

The minimum JCL that is required for execution of Fast Path Offline Suite product functions is simple because many of the necessary data sets can be allocated dynamically. If you supply the appropriate DD statements in your JCL, the specified data sets are used. BMC recommends that you omit these DD statements so that dynamic allocation is used.

When you are using dynamic allocation, Fast Path Offline Suite products search for the appropriate DFSMDA members in the following order:

1. in the data sets that are allocated to the STEPLIB DD statement
2. in the data sets that are allocated in LNKLIST

**IMS ACB library**

Fast Path Offline Suite product functions require access to the IMS ACB library containing the database definitions (DMB) for the DEDB area or areas to be processed. The following JCL DD statements are searched in the order listed for the IMS ACB library:

- IMSACB
- OLCSTAT (IMS version 8.1 or later)
- MODSTAT / MODSTAT2
- IMSACBA
- IMSACBB

If one of the first three DD statements listed above is not included in the JCL, the Fast Path Offline Suite product tries to dynamically allocate it in the following order:

1. STEPLIB/LNKLIST are searched for the OLCSTAT DFSMDA member (IMS version 8.1 or later only). If found, the OLCSTAT data set is interrogated to determine whether IMSACBA or IMSACBB is the active current library.

2. If OLCSTAT DFSMDA is not found, the STEPLIB/LNKLIST are searched for the MODSTAT DFSMDA member. If found, the MODSTAT data set is interrogated to determine whether IMSACBA or IMSACBB is the active current library.

3. If MODSTAT DFSMDA is not found, the STEPLIB/LNKLIST are searched for the MODSTAT2 DFSMDA member. If found, the MODSTAT2 data set is interrogated to determine whether IMSACBA or IMSACBB is the active current library.

4. If the IMSACBA or IMSACBB DD statement is not present in the JCL, the STEPLIB/LNKLIST are searched for the DFSMDA member.
For details about using DD statements to identify the IMS ACB library, see the IMS/ESA System Administration Guide.

**NOTE**
The IMS/ESA release level of the ACB library must be the same as that of the RESLIB that is included in the STEPLIB DD statement concatenation.

**Area data set**

Fast Path Offline Suite product functions require access to the area data set to be processed. The areaname DD statement identifies the area data set to be processed. If the areaname DD statement is omitted from the JCL, the product tries to dynamically allocate it.

The areaname DD statement is the area data set that is used as input to analyze, DMAC print, and unload functions, as output from change and reload functions, and as input and output of reorganize and initialization functions. The IMSACB DD statement identifies the ACB library containing the database definition that describes the area which is referenced by the areaname DD statement.

The areaname DD statement can specify an image copy data set with analyze, DMAC print, and unload functions.

If DBRC is active, the area is registered with DBRC, and the areaname DD statement refers to an area data set, the areaname DD statement data set must match the registered area data set name.

If you are using dynamic allocation, do not include the areaname DD statement. The Fast Path Offline Suite product tries to obtain the data set name for allocation in the following order:

1. If the INPUT_DSN_MASK keyword is specified, it is used to generate the data set name for the analyze, DMAC print, initialize, reorganize, and unload functions.

2. If the OUTPUT_DSN_MASK keyword is specified, it is used to generate the data set name for the change and reload functions.

3. If DBRC is active and the area is registered, the registered area data set name is obtained from DBRC.

4. The STEPLIB/LINKLIST is searched for the DFSMDA member that contains the data set name for this area.
Disposition of the area data set, whether dynamically allocated or specified in the JCL, depends on the command function that is processed, as shown in Table 3:

### Table 3  Area data set disposition for Fast Path Offline Suite command processes

<table>
<thead>
<tr>
<th>Command function</th>
<th>Area data set disposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANALYZE</td>
<td>DISP=SHR</td>
</tr>
<tr>
<td>CHANGE (output)</td>
<td>DISP=OLD</td>
</tr>
<tr>
<td>DMAC_PRINT</td>
<td>DISP=SHR</td>
</tr>
<tr>
<td>EXTEND</td>
<td>DISP=OLD</td>
</tr>
<tr>
<td>INITIALIZE</td>
<td>DISP=OLD</td>
</tr>
<tr>
<td>RELOAD (output)</td>
<td>DISP=OLD</td>
</tr>
<tr>
<td>REORGANIZE</td>
<td>DISP=OLD</td>
</tr>
<tr>
<td>UNLOAD (input)</td>
<td>DISP=SHR</td>
</tr>
</tbody>
</table>

**Using DBRC**

Fast Path Offline Suite products can interact with DBRC to perform several activities. BMC recommends the use of DBRC. Activation and use of DBRC depends on how your IMS system has been defined, and upon the value that is set by the DBRC keyword of the GLOBAL command.

By default, Fast Path Offline Suite products use the value that is set by the IMS system definition:

- If your IMS system has been defined to not use DBRC by default (the IMMSGEN is defined by using DBRC=(YES,NO)), this default is passed to Fast Path Offline Suite products and DBRC is not activated during function processing. To activate DBRC, specify DBRC=YES on the Fast Path/EP GLOBAL command.

- If your IMS system has been defined to use DBRC by default (the IMMSGEN is defined by using DBRC=(YES,YES)), this default is passed to Fast Path Offline Suite products and DBRC is activated during function processing. To bypass DBRC processing, specify DBRC=NO on the GLOBAL command.

- If your IMS system has been defined to force the use of DBRC (the IMMSGEN is defined by using DBRC=FORCE), Fast Path Offline Suite products activate DBRC during function processing. You cannot deactivate DBRC by specifying DBRC=NO on the GLOBAL command. If it is necessary to deactivate DBRC, you can specify DBRC=FORCEOFF on the GLOBAL command, although this setting is not recommended for normal product processing.
DBRC functions

When DBRC is active during the execution of Fast Path Offline Suite product execution, it is used for the following functions:

- **access control**
  
  Fast Path Offline Suite products request access authorization for each area before its use. When the product is finished processing the area, it requests deauthorization of the area. The authorization level that is used for the area depends on the requirements of the product function that is executing.

- **dynamic allocation of area data sets**
  
  Fast Path Offline Suite products use the registered name of the area data sets for dynamic allocation. If multiple area data sets (MADS) are specified for an area, the product uses the first available data set in the ADS list.

- **recovery control**
  
  Fast Path Offline Suite products record events, as necessary, that cause a change in the recovery status of the area.

- **image copy notification**
  
  Fast Path Offline Suite products record the successful creation of image copy data sets.

RECON data sets

If Fast Path Offline Suite product functions use DBRC during processing, they require access to the RECON data sets.

If RECON data sets are not supplied in the execution JCL, they are dynamically allocated by DBRC. DBRC searches through the data sets that are allocated to the STEPLIB/LNKLIST for the DFSMDA members RECON1, RECON2, and RECON3.
Multiple area data sets

Multiple area data sets (MADS) are not supported in offline mode. Fast Path Offline Suite products process only one area data set that is registered in the ADS list for an area.

- For functions that read the database but do not update it, the ADS list that is registered for each area is searched (in collating sequence by DD name) for the first ADS that is marked as available for use and that has no error queue elements (EQEs). The selected ADS is read and processed; all other area data sets are ignored.

- For functions that read and update the database, the ADS list that is registered for each area is searched (in collating sequence by DD name) for the first ADS that is marked as available for use and that has no EQEs. The selected ADS is read and updated; all other area data sets are marked as unavailable.

- For functions that create the database, the first ADS (in collating sequence by DD name) is selected and created. All other area data sets are marked as unavailable.

- When functions that update or create the database have been completed successfully, only one ADS will be marked as available. Use the IBM Online MADS Create utility to resynchronize the other (unavailable) area data sets.

Multitasking

Fast Path Offline Suite products create multiple tasks for performing the requested product functions, and for various internal support activities. Each product command that is specified in the PFPSYSIN control statements is executed as a separate task.

In the example shown in Figure 7, two commands are specified. Because each command is executed as a separate task, analysis of the areas within the dbdname1 occurs in parallel with the reorganization of the areas within dbdname2. Do not specify the same DEDB/area combination on multiple commands in the same command set.

Figure 7  Control statements for separate product tasks

```
ANALYZE DBD=dbdname1, IAREA=ALL
END
REORGANIZE DBD=dbdname2, IAREA=ALL
END
```
Serial processing

When a command specifies that multiple areas are to be processed (both commands in the previous example specify multiple areas), the list of areas is processed serially. If `dbdname1` contains three areas (`area1`, `area2`, and `area3`, for example), `area1` is processed to completion before `area2` begins. In turn, `area2` is processed completely before `area3` begins.

In Figure 8, a single task is performed on multiple areas. The three areas are processed serially in the specified order.

**Figure 8  Control statement for serial tasks**

```
ANALYZE DBD=dbdname1,IAREA=(area1,area2,area3)
```

Parallel processing

In Figure 9, the same three areas are processed as in the previous example. In this example, however, separate tasks are used to analyze the areas in parallel.

**Figure 9  Control statements for parallel tasks**

```
ANALYZE DBD=dbdname1,IAREA=area1
ANALYZE DBD=dbdname1,IAREA=area2
ANALYZE DBD=dbdname1,IAREA=area3
```

Serial and parallel processing

You should consider the availability of various system resources when requesting that processing be performed in parallel. The number of processor engines, the amount of real and virtual storage available (above and below the 16 MB boundary), and other factors affect the relative efficiency of serial processing in comparison to parallel processing.

You can limit the number of command tasks that are started in parallel by using the `PRODUCT_LIMIT` keyword on the `OPTIONS` command.

Caching

Fast Path Offline Suite products provide a caching mechanism that can improve performance in some instances. The caching mechanism is used to process significant portions of the area within virtual storage (using a data space) rather than using physical read and write I/O operations to DASD.
The caching function is most beneficial when the product processes control intervals in a random (non-sequential) manner. The cache is loaded by using a high-speed sequential read process, and unloaded (if necessary) by using a sequential write process. Because random processing is performed within the data space, the time that is required to perform random I/O operations to DASD is significantly reduced. This advantage is offset, however, by the resources that are required to maintain the data space within virtual storage.

It is neither necessary nor beneficial to load the entire area into a cache. The base UOW control intervals (RAA and DOVF blocks), for example, are always processed sequentially. Caching does not provide any benefit for sequential processing. The IOVF and SDEP control intervals can be accessed randomly by some functions; therefore, a potential benefit might result by caching IOVF blocks, SDEP blocks, or both. The potential benefit of caching is described in the “Performance Considerations” or “Enhancing Performance” section in other chapters in this book.

**Input areas**

When caching is requested for an input area (using the ICACHE keyword), portions of the area data set are read sequentially into a data space. As the product processes the cached control intervals, the control intervals are read from or written to the data space rather than DASD. When product processing completes, any control intervals that have been modified within the data space are written sequentially to DASD.

**Output areas**

When caching is requested for an output area (using the OCACHE keyword), portions of the area data set are kept within a data space. (Because an output area is normally empty when the processing begins, it is not necessary to read it from DASD to load the cache.) As the product processes the cached control intervals, the control intervals are read from or written to the data space rather than DASD. When the product completes, the control intervals within the data space are written to DASD sequentially.

**Image copy input**

Certain Fast Path Offline Suite product functions can use an image copy data set as input rather than the actual area data set. Random access to an image copy data set is not available. By loading the IOVF and SDEP control intervals into cache, however, random access is made possible. This action occurs automatically when an image copy data set is used as input to the change and unload functions.
System resource limits

When an area or portions of an area are cached, a data space is created. A data space cannot exceed 2 GB, and must be backed by expanded storage and/or paging space. Fast Path Offline Suite products use a compression technique to minimize the size of the data space, but the total amount of data space storage that is available is frequently limited.

Fast Path Offline Suite products query the operating system about the amount of data space storage that is available. The products do not try to create a data space that would exceed the capacity of the extended storage and page space that is available to the operating system.

Simplifying the command set

Fast Path Offline Suite products provide an extensive and powerful command language. Simple keywords and subcommands are used to request available features and functions. Specifying these keywords and subcommands on each command can become repetitious. You can simplify the command set and reduce tedious repetition of keywords and subcommands by setting global values for many common keywords.

If specified on the GLOBAL command, the value for a keyword is applied globally on all commands in the command set for which the keyword is valid.

The example in Figure 10 shows how to set a value for the HISTORY_DDNAME keyword that applies to all other commands in the set. The commands to analyze dbdname1 and dbdname4 inherit this value as if it had been explicitly specified on the ANALYZE commands. Because the command to analyze dbdname2 contains an explicit specification for the HISTORY_DDNAME keyword, the value abc overrides the value xyz for that database.

Figure 10 Specifying the GLOBAL command

<table>
<thead>
<tr>
<th>Command</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLOBAL</td>
<td>HISTORY_DDNAME=xyz</td>
</tr>
<tr>
<td>ANALYZE</td>
<td>DB=dbdname1</td>
</tr>
<tr>
<td>ANALYZE</td>
<td>DB=dbdname2,HISTORY_DDNAME=abc</td>
</tr>
<tr>
<td>ANALYZE</td>
<td>DB=dbdname4</td>
</tr>
</tbody>
</table>

You can also specify the IC, REPORT, and THRESHOLD subcommands under the GLOBAL command. These subcommands are implicitly applied as the values on all commands in the command set where they are valid.
Randomizer module interface

The example in Figure 11 shows how to specify a REPORT subcommand that applies to all other commands in the set. The commands to analyze dbdname1 and reorganize dbdname3 (parallel analysis) inherit this subcommand as if it had been specified explicitly on the two commands. Because the command to analyze dbdname2 contains an explicit REPORT subcommand of its own, this explicit REPORT subcommand overrides the REPORT subcommand that is specified on the GLOBAL command.

![Figure 11 Setting a report default](image)

The example in Figure 12 shows how to request the reorganization of three DEDBs. During reorganization, full pointer validation is performed and an output image copy is generated for each area.

![Figure 12 Setting a report default](image)

Randomizer module interface

The randomizer module interface environment conforms to the published interface, with the following exceptions:

- When a randomizer is invoked by IMS, register 10 contains the address of the EPST and register 11 contains the address of the ESCD. Some user-written or user-customized randomizers have been designed to use these addresses to gain access to IMS/VSE control blocks other than those that are passed as part of the published interface.

- When a Fast Path Offline Suite product issues a call to a randomizing module, register 10 is set to -1 and register 11 is set to 0 to indicate that the call is not being issued in a live IMS environment.
Fast Path Offline Suite products provide the capability to temporarily or permanently customize eligible messages that are issued by any Fast Path Offline Suite primary command. Eligible refers to messages that are available for customization as defined within the product. Temporarily means that you can specify customizations that will apply only to the job input where the changes are requested. Permanently means that you can specify message customizations that will apply to all subsequent Fast Path Offline Suite job execution until you choose to return customizations to product defaults.

You might find message customization useful for the following situations:

- to reduce the “nuisance factor” of a message, such as the number of times a message about segment length errors is issued for a very large area

- to reduce the severity level of a message from E (error) to W (warning) so that issuance of the message will not terminate processing (such as an error message that would otherwise terminate an unload process)

- to increase the severity level of a condition-specific message to force an automatic snap dump in response to a detected condition within the DEDB or area

Permanent changes that you request for eligible messages are stored in the Fast Path/EP statistics repository and are implemented by coding command language executed by the PFPEPR00 repository program. Changes to messages that are specified in this manner will apply as long as they remain stored in the repository.

When you have defined message customizations and stored them in the repository catalog, you must define the repository catalog data set name in your job input to activate message customizations. If you decide later to undo any (or all) message customizations, you can execute PFPEPR00 to return all (or selected) messages to product defaults.

Temporary changes that you request for eligible messages are specified in the JCL under the PFPOPTS DD statement. After job execution, any specified changes will restore the previous settings automatically.
Permanently modifying messages

Using the PFPEPR00 utility, you can accomplish the following functions for eligible messages that are issued by the Fast Path Offline Suite products:

- override (change) the default suffix level (informational, warning, error or critical)
- suppress issuance of the specified message by all subsequent primary command processes after reaching a specified occurrence threshold
- restore any (or all) customizations to product defaults
- list all active message customizations that you already specified by a previous execution of PFPEPR00

Any customizations that you specify for messages by using the PFPEPR00 utility will apply to all Fast Path Offline Suite primary command processes until you execute PFPEPR00 again to change customizations or restore them to product defaults.

For detailed instructions about using the PFPEPR00 utility to specify, store, list, and remove permanent message customizations, see Appendix B, “Dynamic message modification.”

Temporarily modifying messages

Using the PFPOPTS DD statement, you can combine the OPTIONS command and the OVERRIDE subcommand to customize messages for the job input in the associated PFPSYSIN DD statement:

- override (change) the default suffix level (informational, warning, error or critical)
- suppress issuance of the specified message by the primary command process after reaching a specified occurrence threshold

Table 4 shows the functions of the keywords that are available on the PFPOPTS OVERRIDE subcommand to customize messages for the process that is defined by the associated PFPSYSIN input.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>MESSAGE_LEVEL</td>
<td>change default severity level for message</td>
</tr>
<tr>
<td>MESSAGE_LIMIT</td>
<td>set threshold level for suppression of message</td>
</tr>
<tr>
<td>MESSAGE_NUMBER</td>
<td>specify product ID number of message to be customized</td>
</tr>
</tbody>
</table>
You must specify a separate OVERRIDE subcommand for each message you want to customize. Figure 13 specifies two sets of message customization syntax on the PFPOPTS DD statement. The first OVERRIDE subcommand is used to reduce the severity level of message BMC111162 from E (the default) to W. The second OVERRIDE subcommand suppresses the issuance of message BMC111162 after it has been issued 10 times by the unload process.

**Figure 13** Specifying message overrides on the PFPOPTS DD statement

```plaintext
//PFP      EXEC PGM=PFPMAIN,REGION=0M
//STEPLIB  DD DSN=BMC.PFP.LOAD,DISP=SHR
//       DD DSN=IMSVS.RESLIB,DISP=SHR
//PFPOPTS  DD *
       OPTIONS
           OVERRIDE MESSAGE_NUMBER=111162
           MESSAGE_LEVEL=WARNING
           OVERRIDE MESSAGE_NUMBER=121150
           MESSAGE_LIMIT=10
/*
//PFPSYSIN DD *
UNLOAD DBD=dbdnam1,IAREA=areaname
/*
```

**Temporarily restoring permanent message customizations**

If you have already used the PFPEPR00 utility to set permanent modifications for eligible messages, you can specify the RESET subcommand on the PFPOPTS DD statement to temporarily restore any (or all) customized messages to product defaults. RESET can be used to temporarily override the permanent override that you stored in the Fast Path/EP repository.

For detailed instructions about using the PFPEPR00 utility to specify, store, list, and remove permanent message customizations, see Appendix B, “Dynamic message modification.”

The example in Figure 14 assumes that PFPEPR00 has been executed previously to override the suffix level for message BMC111162 and to set a suppression threshold for message BMC121150. Separate RESET subcommands are specified to restore both messages to product defaults for the specified job input only.

**Figure 14** Restoring selected permanent message overrides on the PFPOPTS DD statement (part 1 of 2)

```plaintext
//PFP      EXEC PGM=PFPMAIN,REGION=0M
//STEPLIB  DD DSN=BMC.PFP.LOAD,DISP=SHR
//       DD DSN=IMSVS.RESLIB,DISP=SHR
//PFPOPTS  DD *
       OPTIONS
```
Suppressing repetitious messages by suffix type

For each anomaly that is encountered in an area, the command function can generate warning, error, informational, or critical messages. When the number of places that a particular condition exists is high, a large number of messages are produced.

You can use the MESSAGE_SUPPRESSION keyword on the primary command to set a threshold level for all messages with a certain suffix. You can reduce the number of repetitious messages by specifying a threshold value on the MESSAGE_SUPPRESSION keyword. Use this keyword to specify the maximum number of times that any warning or error message is to be produced before it is suppressed.

In the example in Figure 16, the MESSAGE_SUPPRESSION keyword specifies that any warning message is to be issued no more than 10 times, that any error message is to be produced no more than 15 times, and that any informational message will be issued no more than five times by the REORGANIZE process.
Suppressing repetitious messages by suffix type

**Figure 16 Using the MESSAGE_SUPPRESSION keyword**

```
REORGANIZE DBD=dbdname, IAREA=areaname,
  MESSAGE_SUPPRESSION=(WARNING=10, ERROR=15, INFORMATIONAL=5)
```

When messages are suppressed in this way, the Fast Path Offline Suite product produces a summary of the number of times that each message has been suppressed.

---

**WARNING**

The MESSAGE_LIMIT keyword takes precedence over the MESSAGE_SUPPRESSION keyword. If you specify a suppression threshold for a specific message by using the OVERRIDE subcommand (with the MESSAGE_LIMIT keyword) on the PFPOPTS DD statement, the product complies with this threshold, even if you specify a different threshold for all messages with the same suffix level (with the MESSAGE_SUPPRESSION keyword) on the primary command.
This chapter provides information on the capabilities, setup, and use of the offline reorganization function that is provided by Fast Path Reorg/EP. This function lets you perform space reclaim and other tasks on DEDBs without database unloads and reloads. This chapter discusses the following topics:

Reorganization function overview .................................................. 62
  Selecting UOWs with patented intelligent reorg .......................... 62
  User-specified control of the reorganization .............................. 63
  Automatic pointer validation during reorganization .................... 63
Reorganization function restrictions .............................................. 63
Reorganization function inputs and outputs .................................. 64
Offline reorganization considerations ......................................... 65
  Control statement ................................................................. 65
  Performance considerations .................................................... 65
  DBRC considerations ............................................................ 67
  Shutting down the reorganization function ............................... 67
  MADS considerations ........................................................... 68
REORGANIZE command keywords and subcommands ..................... 68
Selecting the database and areas ................................................ 69
Allocating the area data set ....................................................... 70
Setting UOW selection criteria ................................................... 71
  Selectively specifying UOWs for reorganization ....................... 71
  Selecting all UOWs for reorganization ................................... 73
Controlling segment compression ................................................. 73
Extending IOVF and SDEP storage portions .................................. 74
  Extending IOVF ................................................................. 75
  Extending SDEP ............................................................... 77
Placing segments with load control ............................................. 78
  Defining segments for load control placement ......................... 79
  Placement control using segment content ................................ 81
  Placement control using segment counts ................................ 82
Controlling error tolerance ........................................................ 83
Analyzing the DEDB during reorganization .................................. 84
Making an image copy during reorganization ............................... 85
Sample offline REORGANIZE command scenarios ....................... 85
Reorganization function overview

In many cases, the need to reorganize an area does not require restructuring the area, but simply the elimination of fragmented free space elements (FSEs) and scraps in the root addressable area (RAA), and the movement of as many segments as possible from independent overflow (IOVF) into their respective RAA or DOVF control intervals (CIs). The reorganization function of Fast Path Reorg/EP offers a high-performance alternative for reclaiming fragmented space in a unit of work (UOW) when compared to the traditional reorganization process of unloading and reloading the DEDB area. The reorganization function, which is executed by the REORGANIZE command, effectively takes advantage of any free space in the UOW and frees as many IOVF CIs as possible with a minimum of I/O. The reorganization function performs the following tasks:

- reclaims space
- resequences roots
- controls segment placement on the reorganized DEDB
- optionally extends the size of the IOVF and SDEP portions of an area during space reclaim without changing the DBD

Selecting UOWs with patented intelligent reorg

Certain characteristics of a UOW can determine whether reorganization of the UOW might result in significant performance improvement. Fast Path Reorg/EP incorporates the patented Intelligent Reorg technology. Intelligent Reorg selects UOWs based on certain user-specified criteria and reorganizes only those UOWs whose characteristics meet the specified criteria. You can specify criteria to answer the following questions:

- Does the UOW extend into IOVF?
- Is UOW fragmentation over a specified threshold?
- Can reorganization of the UOW save a specified amount of IOVF?

High performance is achieved by reorganizing only UOWs that meet the criteria, which saves time and resources during space reclaim maintenance.

Under ordinary conditions, not all UOWs in an area extend into and use IOVF CIs. The Intelligent Reorg feature selects only those UOWs that use IOVF, giving the greatest return with the least I/O to the area. The Intelligent Reorg feature can be turned off, allowing all UOWs to be reorganized, even if the UOW does not extend into IOVF.
Intelligent Reorg examines each UOW to determine whether it is disorganized so that only UOWs that require reorganization are reorganized. This feature reduces I/O, processing, and downtime. To efficiently perform space reclaim, the Intelligent Reorg function follows these steps:

1. reads each UOW of the selected area or areas
2. examines the UOW to determine level of disorganization
3. reorganizes the UOW only if the UOW is disorganized
4. rewrites the UOW to the same data set only if more than a specified amount of IOVF space was reclaimed

User-specified control of the reorganization

User-specified keywords let you control the DEDB reorganization process manually. You can reorganize all areas or selected areas, and you can reorganize all UOWs or selected UOWs by using selected keywords with the REORGANIZE command.

Automatic pointer validation during reorganization

If you have a license for the Fast Path Analyzer/EP product, the product performs quick (checksum) pointer validation by default when you execute the reorganization function.

Reorganization function restrictions

The following restrictions apply to the reorganization function:

- The SELECT_UOW keyword enables you to specify the technique to be used by the reorganization function in the selection of UOWs to be reorganized. The normal default for this keyword is SELECT_UOW=IOVF, which causes reorganization of only UOWs that extend into IOVF. Under the following conditions, however, the reorganization function will automatically use a value of SELECT_UOW=ALL:
  - when compression is requested (either by specifying COMPRESS=YES or COMPRESS=segment name)
— when the EXTEND_IOVF_#UOWS keyword is specified and SDEP segments are defined in the DBD

- When either or both of the conditions specified above necessitate the assignment of SELECT_UOW=ALL, the reorganization function will ignore any value that you specify on the FRAGMENTATION_PERCENT keyword and any alternate value that you specify on the SELECT_UOW keyword. Under either of these conditions, Fast Path Reorg/EP will issue message BMC11119I to indicate that all UOWs will be processed.

- Multiple area data sets (MADS) are not supported for offline reorganizations.

### Reorganization function inputs and outputs

The reorganization function reorganizes the area data set in place, UOW by UOW. For this reason, the same area data set is used as input to (and output from) the reorganization function.

Possible inputs and outputs for an offline reorganization function are shown in Figure 17.

**Figure 17  **Reorganization function inputs and outputs
The reorganization function always issues a summary report to the report log when reorganization activities are complete. For an example of the Reorganization Report, see the PFPREORG member in the REPORTS data set.

### Offline reorganization considerations

Performing an offline reorganization requires that the DEDB be taken offline from IMS.

#### Control statement

The PFPSYSIN control statements include the command set necessary to run the offline reorganization function. A sample control statement for an offline reorganization is shown in Figure 18. Reorganization will be limited to only the area that is specified with the IAREA keyword. Dynamic allocation of the area data sets and the ACB library is assumed.

#### Figure 18   Sample control statement for offline reorganization

```
//PFP EXEC PGM=PFPMAIN,REGION=0M
//STEPLIB   DD   DSN=BMC.PFP.LOAD,DISP=SHR
//      DD   DSN=IMS.RESLIB,DISP=SHR
//PFPSYSIN DD   *
   REORGANIZE DBD=dbdname,IAREA=areaname
/*
```

### Performance considerations

The offline reorganization function has certain performance considerations.

#### Buffer usage

The offline reorganization function automatically determines the number of buffers that will be required before reorganization. This automatic determination of buffer requirements is an advantage over other reorganization utilities. Other utilities require the user to estimate the number of buffers that are required before reorganization and will fail if the estimate is insufficient.
Performance considerations

Analyzing the area during reorganization

If you have a license for the Fast Path Analyzer/EP product, quick (checksum) pointer validation will be performed by default during the reorganization function. The I/O that is required to read the UOW and IOVF control intervals is shared between the two functions.

Because the reorganization process does not access SDEP control intervals, additional I/O is required to read these control intervals if you request SDEP pointer validation. This I/O, however, can be shared with the image copy function.

For more information, see “Analyzing the DEDB during reorganization” on page 84. For example, you can control the level of pointer validation or turn pointer validation off, depending on your performance considerations.

Creating an output image copy during reorganization

If you create one or more output image copies while performing a reorganization, the I/O that is required to read the UOW control intervals is shared between the two functions.

Unlike the reorganization process, the image copy process requires IOVF control intervals to be processed sequentially. Consequently, the I/O that is required to read the IOVF control intervals for the image copy is not shared. The I/O that is required for processing of the SDEP control intervals is shared between image copy and analysis functions.

When the process is completed, the reorganization function notifies DBRC that an image copy is required for each area by setting the “Image Copy Needed” flag. You can use the IC subcommand to request that an image copy be created for each area during the reorganization. DBRC is informed (NOTIFY.IC) when this image copy is created successfully, which resets the “Image Copy Needed” flag.

If you do not take an image copy during the reorganization process, a message is issued to indicate that an image copy must be taken before each area can be updated online.

NOTE

BMC recommends that a valid image copy of the input area exists before you execute an offline reorganization. You should take an image copy of the output area during or after executing the REORGANIZE command.
Message suppression

For each anomaly that is encountered in an area, the reorganization function generates a message with a specific suffix (severity) level. When the number of places that a particular condition exists is large, a large number of messages is produced. You can reduce the number of repetitious messages produced by using the MESSAGE_SUPPRESSION keyword to specify the maximum number of times that an informational, warning, error or critical message is to be produced. For the reorganization process, this keyword functions in the same manner as it does for the Fast Path/EP analysis process. For more information, see “Suppressing repetitious messages” on page 111.

DBRC considerations

When DBRC is active during the execution of the offline reorganization function, and the area is registered with DBRC, an exclusive access (EX) authorization level is requested for the area data set. The data set name for the area data set must match the name that is registered with DBRC.

NOTE

Because the Fast Path Reorg/EP reorganization function does not initialize the DEDB to complete the process, a REORG record is not created in DBRC when the reorganization function executes.

Shutting down the reorganization function

When you are running an offline reorganization, you can terminate the reorganization function before completion. Use the operator interface to enter the SHUTDOWN command. Database integrity cannot be guaranteed if termination of the reorganization function occurs any other way. Normal backup procedures should be followed before executing the reorganization function to provide recovery capability if it is required.

Because the reorganization function reorganizes one UOW at a time, reorganization that has already been performed does not have to be repeated if the reorganization function is interrupted. UOWs that have already been reorganized remain reorganized. After restart, the reorganization function begins processing at the next disorganized UOW. Special restart JCL is not required.
MADS considerations

Multiple area data sets (MADS) are not supported for offline reorganizations. The offline reorganization function searches the area data set (ADS) list that is registered for each area (in collating sequence by DD name). The product selects the first ADS that is marked as available for use and that has no error queue elements (EQEs). If an ADS is found that meets both of these criteria, it is the only ADS that will be reorganized. All other area data sets are marked as unavailable. After the reorganization completes, use the IBM Online MADS Create utility to resynchronize the other (unavailable) area data sets.

If no area data set is marked as available, or if all available area data sets contain one or more EQEs, reorganization is not performed for that area.

REORGANIZE command keywords and subcommands

Table 5 lists the keywords and subcommands that are available for the REORGANIZE command.

<table>
<thead>
<tr>
<th>Function</th>
<th>Command or subcommand</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>selecting the database and area</td>
<td>REORGANIZE DBD IAREA</td>
<td></td>
</tr>
<tr>
<td>allocating the area data set</td>
<td>REORGANIZE INPUT_DSN_MASK</td>
<td></td>
</tr>
<tr>
<td>setting UOW selection criteria</td>
<td>REORGANIZE SELECT_UOW FRAGMENTATION_PERCENT IOVF_SAVE_THRESHOLD</td>
<td></td>
</tr>
<tr>
<td>controlling segment compression</td>
<td>REORGANIZE COMPRESS</td>
<td></td>
</tr>
<tr>
<td>extending IOVF and SDEP storage portions</td>
<td>REORGANIZE EXTEND_IOVF/#UOWS EXTEND_SDEP/#CIS</td>
<td></td>
</tr>
<tr>
<td>placing segments with Load Control</td>
<td>LOADCTL all associated keywords</td>
<td></td>
</tr>
<tr>
<td>controlling error tolerance</td>
<td>REORGANIZE ERROR_THRESHOLD</td>
<td></td>
</tr>
</tbody>
</table>

Table 5 REORGANIZE command keywords and subcommands (part 1 of 2)
Selecting the database and areas

Table 5  REORGANIZE command keywords and subcommands (part 2 of 2)

<table>
<thead>
<tr>
<th>Function</th>
<th>Command or subcommand</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>analyzing the DEDB while reorganizing</td>
<td>REORGANIZE REPORT THRESHOLD</td>
<td>POINTER_VALIDATION RAP_VALIDATION SDEP_VALIDATION LARGEST_DATABASE_RECORDS all associated keywords</td>
</tr>
<tr>
<td>making an image copy of a DEDB while reorganizing</td>
<td>IC</td>
<td>all associated keywords</td>
</tr>
</tbody>
</table>

For more information about the following topics, see the Fast Path/EP Series Reference Manual:

- syntax of commands, subcommands, and keywords that are discussed in this book
- diagrams that show the syntax and available parameters and values for Fast Path Offline Suite commands and subcommands

Selecting the database and areas

The DBD keyword identifies the name of the DEDB (DBD name) to be reorganized. The DBD keyword is required for offline reorganization processing.

The IAREA keyword can be used to select specific areas to be reorganized. If you omit the IAREA keyword, all areas defined in the DEDB are reorganized.

Areas can be specified on the IAREA keyword by using any combination of area names, area numbers, or area ranges. The following parameters are available for the IAREA keyword:

- IAREA=ALL (default) or IAREA=* specifies all areas of the DEDB.
- IAREA=areaname specifies one or more areas by using the one-character to eight-character area name for each area specified. Multiple area names must be enclosed in parentheses and separated by commas.
- IAREA=areanumber specifies one or more areas by using the one-character to five-character area number for each area specified. Multiple area numbers must be enclosed in parentheses and separated by commas.
Allocating the area data set

- IAREA=(RANGE=(startarea,endarea)) specifies a consecutive range of areas using either areaname or areanumber parameters. The area number associated with startarea must be less than the area number associated with endarea.

An asterisk (*) can be used to specify all areas of the DEDB. When the * character is used with the RANGE keyword, it can be used to specify the beginning or ending range for specific areas of the DEDB.

To reorganize an entire DEDB, use a command set like the example shown in Figure 19.

**Figure 19  Sample control statement for reorganizing all areas**

```
REORGANIZE DBD=dbdname,IAREA=ALL
```

To reorganize specific areas, use a command set like the example shown in Figure 20.

**Figure 20  Sample control statement for reorganizing specific areas**

```
REORGANIZE DBD=dbdname,IAREA=(AREANAM1,AREANAM3,RANGE=(5,8))
```

This control statement uses a combination of area names and area numbers to request that the specified areas are to be reorganized. The UOWs in areas named AREANAM1 and AREANAM3 will be reorganized. Also, all consecutive areas from area number 5 to area number 8 (area5, area6, area7 and area8) will be reorganized.

Allocating the area data set

For an offline reorganization, the area data set can be supplied in the JCL or can be accessed by using dynamic allocation. The areaname DD statement identifies the area data set to be reorganized. If the areaname DD statement is omitted from the JCL, Fast Path Reorg/EP tries to dynamically allocate it.

The IMSACB DD statement identifies the ACB library containing the database definition that describes the area that is referenced by the areaname DD statement.

If you are using dynamic allocation, do not include the areaname DD statement. Fast Path Reorg/EP tries to obtain the data set name for allocation in the following order:

1. If the INPUT_DSN_MASK keyword is specified, it is used to generate the data set name.

2. If DBRC is active and the area is registered, the registered area data set name is obtained from DBRC.
3. The STEPLIB is searched for the DFSMDA member that contains the data set name for this area.

If DBRC is active and the area is registered with DBRC, the allocated data set name must match the registered data set name regardless of how it is allocated.

Setting UOW selection criteria

You can control which UOWs are selected for reorganization by using the SELECT_UOW keyword. Available values for this keyword are discussed in the following sections.

Selectively specifying UOWs for reorganization

SELECT_UOW=IOVF (the default) selects UOWs based on IOVF usage. When this value is specified, the reorganization function will identify and reorganize only UOWs that extend into IOVF. By limiting the scope of reorganization, SELECT_UOW=IOVF significantly reduces processing I/O. In the example shown in Figure 21 for database dbdname, the reorganization function will consider UOWs for reorganization in area areaname only if the UOW extends into IOVF.

Figure 21 Sample control statements for selective UOW reorganization

```
REORGANIZE DBD=dbdname, IAREA=areaname, SELECT_UOW=IOVF
```

SELECT_UOW=(IOVF, n1) selects UOWs based on a minimum specified number of IOVF control intervals (CIs) used. When this value is selected and you specify a numeric value for n1, the reorganization function will identify and reorganize only UOWs that have used at least that many IOVF CIs. This value can be used to exclude UOWs from the reorganization that have used a minimum amount of IOVF based on your assessment of CI usage. An example is shown in Figure 22. For database dbdname, the reorganization function will consider UOWs for reorganization in area areaname only if the UOW has used at least five IOVF CIs.

SELECT_UOW=(IOVF, n1, n2) selects UOWs based on a minimum and maximum (optional) specified number of IOVF control intervals (CIs) used. When this value is selected and you specify a numeric value for n1 and n2, the reorganization function will identify and reorganize only UOWs that have used at least the minimum (n1), and no more than the maximum (n2) number of IOVF CIs. This value can be used to exclude UOWs from the reorganization that have used a minimum and maximum amount of IOVF CIs based on your assessment of CI usage.
Either of the preceding values for the SELECT_UOW keyword can be combined with additional keywords to further limit UOW selection criteria.

The FRAGMENTATION_PERCENT keyword is used to specify the percentage of disorganization that must be detected in a UOW that extends into IOVF to qualify the UOW for reorganization.

An example of the FRAGMENTATION_PERCENT keyword is shown in Figure 23. For DBD dbname, the command set reorganizes UOWs in areas areanam1 and areanam2 if the UOW extends into IOVF and the minimum user-specified percent of fragmentation, (2 percent) is detected. Reasonable fragmentation percent values range from 1 to 5 percent. Fragmentation greater than 5 percent is unlikely under ordinary circumstances.

The IOVF_SAVE_THRESHOLD keyword specifies the minimum amount of IOVF that must be saved before the UOW is selected for reorganization. The amount can be specified as the number of CIs or as a percentage of the IOVF that is used by the UOW. Figure 24 specifies that UOWs will not be reorganized unless five or more IOVF CIs will be released.
Selecting all UOWs for reorganization

SELECT_UOW=ALL selects all UOWs for reorganization. This keyword and value specify that all UOWs in an area be selected for reorganization and all will be rewritten. Reorganizing all UOWs increases your reorganization execution time, but results in an area with no fragmentation.

NOTE
Because certain conditions require that all UOWs be processed, the reorganization function will automatically use a value of SELECT_UOW=ALL under the following conditions:

- when compression is requested (either by specifying COMPRESS=YES or COMPRESS=segment name)
- when the EXTEND_IOVF_#UOWS keyword is specified and SDEP segments are defined in the DBD

An example for SELECT_UOW=ALL is shown in Figure 26. For database dbdname, all UOWs in area areaname are reorganized.

Controlling segment compression

If the BMC DATA PACKER/IMS product is installed in the database that you are reorganizing, you can use the COMPRESS keyword to dynamically expand and compress segment data during the reorganization. SDEPs are excluded because the REORGANIZE command does not process SDEP segments.
The COMPRESS keyword feature lets you change compression techniques or other DATA PACKER/IMS product parameters. For more information, see the DATA PACKER/IMS User Guide or contact BMC DATA PACKER/IMS technical support.

The following values are available for the COMPRESS keyword:

- COMPRESS=NO (the default) causes segment data to be left unchanged.
- COMPRESS=YES causes compressed segment data to be expanded then recompressed.
- COMPRESS=(segment name1, segment name2, ..., segment namen) causes only the segments which are listed by name to be expanded then recompressed.

**NOTE**
When COMPRESS=YES or COMPRESS=(segment name) is specified, the offline reorganization function automatically uses the keyword value SELECT_UOW=ALL to ensure that all segments are expanded and recompressed.

In the example in Figure 27, IAREA defaults to ALL. The segments that have a compression exit specified in the DBD are expanded and recompressed.

**Figure 27 Using the COMPRESS keyword with the REORGANIZE command**

```plaintext
REORGANIZE DBD=dbdname,COMPRESS=YES
```

**Figure 28** shows an example of using the COMPRESS keyword to selectively specify segment compression for specified area. segment1, segment4 and segment5 in area1 are expanded and recompressed only if these segments have a compression exit specified in the DBD. Any other (unspecified) segments that have a compression exit specified in the DBD remain unchanged.

**Figure 28** Selectively specifying segments for compression with reorganize

```plaintext
REORGANIZE DBD=dbdname,IAREA=area1,
COMPRESS=(segment1,segment4,segment5)
```

**Extending IOVF and SDEP storage portions**

If an out-of-space condition occurs in IOVF or SDEP, you can use the Fast Path Reorg/EP patented Extend feature to extend the IOVF and SDEP portions of a DEDB during a reorganization. The ability to extend the IOVF and SDEP portions of the area during the reorganization process can save time and resources in comparison to using the CHANGE or UNLOAD/RELOAD commands.
Extending IOVF

Depending on how many IOVF or SDEP portions you request, the number of control intervals (CIs) that are required to accommodate your request is rounded up to the next control area (CA) boundary. If additional CIs must be added as a result of the rounding, the additional CIs are added to the SDEP portion of the database.

NOTE
If you specify the EXTEND_IOVF_#UOWS keyword on the REORGANIZE command and SDEP segments are defined in the DBD, the offline reorganization function will automatically use the keyword value SELECT_UOW=ALL (instead of the standard default SELECT_UOW=IOVF). Processing of all UOWs ensures that all SDEP pointers are updated to accommodate the IOVF extension, but might increase processing time for the reorganization.

The following sections present step-by-step procedures for extending IOVF, extending SDEP, and extending IOVF or SDEP without reorganizing the area.

Extending IOVF

The IOVF portion is extended in “UOW units” of CIs. The optional EXTEND_IOVF_#UOWS keyword allows the REORGANIZE command to extend the IOVF portion. When the IOVF increases by the specified number of UOWs, the SDEP CIs (if SDEP segments are defined and present), are moved to accommodate the increase.

To extend the IOVF during reorganization

1 Ensure that adequate space is present on the primary volume of the target area data set to accommodate the request before attempting to increase the IOVF portion during the reorganization function. If adequate space is not present, allocate additional volumes to the area data set by using IDCAMS ALTER ADDVOLUME.

2 Run the reorganization function by using the EXTEND_IOVF_#UOWS keyword to specify the number of UOWs to add.

3 When the reorganization function is complete, message BMC111193I is issued to the processing log, showing the new ROOT parameter for the DBD source. Use the values that are indicated in message BMC111193I to update the DBD definition to include the IOVF extension. Message BMC110000I is also issued, stating that the extension has been committed.
Sample control statements for extending IOVF

Sample control statements for extending IOVF during reorganization are shown in Figure 29.

**Figure 29  Sample control statements for extending IOVF during reorganization**

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>REORGANIZE DBD=dbdname, IAREA=areaname, EXTEND_IOVF_#UOWS=50</td>
</tr>
</tbody>
</table>

Sample output

The following example shows messages BMC111193I and BMC110000I that are generated when adding 50 UOWs to the IOVF portion:

BMC111193I <dbdname>, <areaname>: ROOT parameter changed from (200,100) to (250,150)

BMC110000I <dbdname>, <areaname>: AREA extension committed <date>, <time>

When the database has been started for IMS, the following message, DFS3703I, is issued by IMS in the control region the first time that the area is opened:

DFS3703I Size of independent overflow has been increased for area <areaname>

Subsequent maintenance on the DEDB

When the extend for IOVF has been committed, the ACB is not updated with the increased ROOT parameter. You must update the DBD and the ACB with the increased ROOT size.

If the ACB is not updated and the area is ever recovered, reloaded, or reinitialized to a smaller sized IOVF, IMS will refuse to bring the area data set online.

To ensure that the extended IOVF is applied to the DBD definition and that the ACB definition is revised

1. Update the ROOT parameter of the AREA statement in the DBD source with the values that you recorded from message BMC111193I.

2. Run DBDGEN.

3. Run ACBGEN.
Extending SDEP

The SDEP storage portion can be extended in CIs. To extend the SDEP storage area, use the optional EXTEND_SDEP_#CIS keyword with the REORGANIZE command. Specify the number of CIs to add. When SDEP storage is increased, the additional CIs are added to the end of the area.

To extend SDEP storage

1. Ensure that adequate space is present on the primary volume of the target area data set to accommodate the request before trying to increase the SDEP portion during the reorganization function. If adequate space is not present, allocate additional volumes to the area data set by using IDCAMS ALTER ADDVOLUME.

2. Before running the reorganization function with the EXTEND_SDEP_#CIS keyword, BMC recommends that you execute the SDEP Scan/Delete utilities.

   **NOTE**
   If the SDEP storage area is in a logically “wrapped” condition when the extend is performed, the new CIs will appear to be “in use” (even though they are empty) until you execute the SDEP Scan/Delete utilities.

3. Run the reorganization function with the EXTEND_SDEP_#CIS keyword to specify the number of CIs to add.

Sample control statement for extending SDEP during reorganization

A sample control statement for extending SDEP during reorganization is shown in Figure 30.

**Figure 30** Sample control statements for extending SDEP during reorganization

```plaintext
REORGANIZE DBD=dbdname, IAREA=areaname,
          EXTEND_SDEP_#CIS=100
```
Sample output

The following example shows message BMC110000I that is generated when adding control intervals to the SDEP portion:

BMC110000I <dbname>, <areaname>: AREA extension committed <date>, <time>

Subsequent maintenance on the DEDB

The next time that you perform maintenance on the DEDB, you might need to change your area data set VSAM cluster definition to allow for the extended size of the SDEP portion.

Placing segments with load control

The normal space search algorithm places segments into the control intervals in a tightly packed manner. The root segment is placed within the RAA block (if room is available) that is determined by the randomizer. Direct dependent segments belonging to the root are placed into the same RAA block or DOVF blocks within the UOW until they are all used. IOVF blocks are selected and filled as needed.

Placement of segments within the types of control intervals (RAA, DOVF, and IOVF) has a significant impact on performance. Consequently, it is sometimes useful to influence the space search algorithm during the reorganization function to relegate infrequently used segments into IOVF (or DOVF) blocks. Segment placement will preserve space in the RAA (or DOVF) blocks for more active data.

Placement of selected segments during the reorganization function can be accomplished by using the LOADCTL subcommand and its related keywords. The LOADCTL subcommand lets you control segment placement in the following ways:

- select a segment, a segment and its dependents, or its dependents only for placement
- control the placement of selected segments into IOVF or DOVF
- control placement of segments, based on their data content
- control placement of segments, based on a specified number of segment occurrences
NOTE

To maintain this data placement, you must continue to use the same LOADCTL subcommand (or subcommands) for any subsequent reorganization processing on the DEDB.

Figure 31 shows a typical DEDB hierarchy. This hierarchy is used in subsequent examples in this section that show how the LOADCTL subcommand can be used with its associated keywords.

Figure 31  Database hierarchical structure in LOADCTL examples

---

Defining segments for load control placement

The SEGMENT keyword is required with the LOADCTL subcommand to identify segments for load control placement. The following subparameters are available for the SEGMENT keyword to provide selection versatility:

- **SEGMENT=(segname,ONLY) (the default) specifies that the LOADCTL subcommand applies to the named segment only. Its dependent segments (if any) are not affected.**

- **SEGMENT=(segname,DEPENDENTS) specifies that the LOADCTL subcommand applies to the dependents of the named segment only. The named segment is not affected.**

- **SEGMENT=(segname,BOTH) specifies that the LOADCTL subcommand applies to the named segment and all its dependents.**

The LOCATION keyword specifies where you want the segments placed. The following values are available:

- **LOCATION=IOVF (the default) places segments in IOVF storage regardless of the amount of space that is available in RAA and/or DOVF blocks.**
Defining segments for load control placement

- LOCATION=DOVF places segments in DOVF storage, regardless of the amount of space available in the RAA block. If an inadequate amount of space exists in the DOVF blocks, the segment is placed in IOVF.

**Placing a specific segment type**

If you specify the SEGMENT keyword with the ONLY subparameter (the default), the named segment only is placed by using load control. For example, in Figure 32, all occurrences of SEGB are placed in IOVF.

**Figure 32  Sample LOADCTL with SEGMENT only**

<table>
<thead>
<tr>
<th>REORGANIZE DBD=dbdname</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOADCTL SEGMENT=(SEGB,ONLY),LOCATION=IOVF</td>
</tr>
</tbody>
</table>

**Placing dependents of a segment type**

If you specify the SEGMENT keyword with the DEPENDENTS subparameter, all dependents of the named segment are placed by using load control. The named segment is placed normally. In the example in Figure 33, the DEPENDENTS subparameter is specified. Segment placement control is applied to all dependent segment types of the named segment (SEGB). All dependent segment occurrences (SEGC, SEGD, SEGE, and SEGF) are placed within an IOVF block, regardless of the amount of space that might remain within the RAA and DOVF blocks.

**Figure 33  Sample LOADCTL for segment dependents**

<table>
<thead>
<tr>
<th>REORGANIZE DBD=dbdname,IAREA=areaname</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOADCTL SEGMENT=(SEGB,DEPENDENTS)</td>
</tr>
</tbody>
</table>

**Placing a segment and its dependents**

If you specify the SEGMENT keyword with the BOTH subparameter, the named segment and all its dependents are placed by using load control.

In the example in Figure 34, the BOTH subparameter is specified. Segment placement control is applied to SEGG and its dependent segment (SEGH).

**Figure 34  Sample LOADCTL for segment and its dependents**

<table>
<thead>
<tr>
<th>REORGANIZE DBD=dbdname,IAREA=areaname</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOADCTL SEGMENT=(SEGG,BOTH),LOCATION=DOVF</td>
</tr>
</tbody>
</table>
You can specify that placement control is to be applied, based on the data values contained in segments by specifying the WHERE keyword. The WHERE keyword lets you specify field names or field positions and lengths, an operand, a value, and optional Boolean operators. For details about the operations and syntax of the WHERE keyword, see the Fast Path/EP Series Reference Manual. For detailed syntax rules for coding these expressions, see the Fast Path/EP Series Reference Manual.

You can specify that placement control is to be applied, depending on values that appear within the named segment data. The example in Figure 35 requests that placement control be applied to the segment type that is specified explicitly by the SEGMENT keyword. Selection criteria is specified by using the WHERE keyword. If the criteria is met, SEGB is placed within an IOVF block, regardless of the amount of space that might remain within the RAA and DOVF blocks. If the criteria is not met, the segment occurrence is placed normally.

You can specify that criteria be based on data within a different segment with the WHERE keyword by using a qualified field reference. The qualified field must lie within the segment parentage path from the root segment to the segment that is specified by using the SEGMENT keyword.

The example in Figure 36 requests that placement control be applied to SEGD that is specified by the SEGMENT keyword. Selection criteria is specified by using the WHERE keyword that references a field within the parent segment (SEGB). If the criteria is met, the SEGD segment occurrences are placed within an IOVF block, regardless of the amount of space that might remain within the RAA and DOVF blocks. If the criteria is not met, the SEGD segments are placed normally.
Placing multiple segments based on different criteria

You can specify as many LOADCTL subcommands as are needed to obtain the desired segment placement.

The example in Figure 37 requests that placement control be used for four separate segments. The WHERE keyword is used for each segment to specify a Boolean expression that controls the placement of each segment occurrence.

Figure 37 Using multiple LOADCTL subcommands to control segment placement

```
REORGANIZE DBD=dbdname,IAREA=areaname
LOADCTL SEGMENT=SEGB,LOCATION=IOVF,
    WHERE=(3:2 EQ C'80'
         OR 3:2 EQ C'90')
LOADCTL SEGMENT=SEGF,LOCATION=IOVF,
    WHERE=(SEGA.16:1 EQ C'N')
LOADCTL SEGMENT=SEGG,LOCATION=IOVF,
    WHERE=(16:4P LE P'19970101' AND
           20:1 EQ C'Y')
LOADCTL SEGMENT=SEGI,LOCATION=IOVF,
    WHERE=(13:2X GT X'4040')
```

If multiple LOADCTL subcommands are specified for the same segment type, they are evaluated in the order in which they are specified. The first LOADCTL subcommand for the segment for which the WHERE criteria is met (or if no WHERE criteria is specified) will be used to control placement.

Figure 38 shows an example of multiple LOADCTL subcommands that are specified for the same segment. If the first WHERE criteria is met, SEGB is placed in IOVF. If the second WHERE criteria is met, SEGB is placed in DOVF. If neither WHERE criteria is met, the segment is placed normally.

Figure 38 Sample LOADCTL using multiple WHERE statements for the same segment type

```
REORGANIZE DBD=dbdname,IAREA=areaname
LOADCTL SEGMENT=SEGB,WHERE=(3:2 EQ '80'),
    LOCATION=IOVF
LOADCTL SEGMENT=SEGB,WHERE=(3:2 GE '80'),
    LOCATION=DOVF
```

Placement control using segment counts

You can specify that placement control is to be applied after a number of segment occurrences for a root segment have been processed by using the INSERT_LIMIT_COUNT keyword.
The example in Figure 39 requests that placement control be used for a specified segment type. The first five segments of this type for each root segment are placed normally. The sixth occurrence (and all subsequent occurrences) of the segment type are placed within an IOVF block, regardless of the amount of space that might remain within the RAA and DOVF blocks within the UOW.

**Figure 39**  Sample LOADCTL using segment occurrences

```plaintext
REORGANIZE  DBD=dbdname,IAREA=areaname
     LOADCTL SEGMENT=SEGE,LOCATION=IOVF,
            INSERT_LIMIT_COUNT=5
```

The example in Figure 40 combines WHERE and INSERT_LIMIT_COUNT keywords on a single LOADCTL subcommand. All segment types that do not meet the WHERE criteria are placed normally. The first three segments that do meet the WHERE criteria are placed normally. The fourth occurrence (and all subsequent occurrences) of the segment type that meet the WHERE criteria are placed within an IOVF block, regardless of the amount of space that might remain within the RAA and DOVF blocks within the UOW.

**Figure 40**  Sample LOADCTL using segment occurrences and qualified selection criteria

```plaintext
REORGANIZE  DBD=dbdname,IAREA=areaname
     LOADCTL SEGMENT=SEGE,LOCATION=IOVF,
            WHERE=(3:2 EQ C'80'),
            INSERT_LIMIT_COUNT=3
```

### Controlling error tolerance

A UOW can contain a pointer error that prevents the reorganization function from processing root or dependent segment chains. If such an error is encountered in a UOW, the reorganization function terminates.

The ERROR_THRESHOLD keyword can be used to enable the reorganization function to encounter and bypass a specified number of UOWs containing pointer errors without terminating. When ERROR_THRESHOLD is used with the REORGANIZE command, any UOW where a pointer error is detected is not reorganized (remains unchanged). Processing continues with the next eligible UOW. When the number of UOWs where errors are encountered exceeds the value specified on the ERROR_THRESHOLD keyword, processing of the area terminates.

When the default value of 0 (zero) is in effect for the ERROR_THRESHOLD keyword, reorganization of the area terminates when the first pointer error is encountered.
The example in Figure 41 requests a reorganization and will bypass up to three UOWs containing pointer errors. The UOWs that contain pointer errors are not reorganized. If a fourth UOW that contains a pointer error is encountered, the reorganization is terminated.

Figure 41  Using the ERROR_THRESHOLD keyword to control error tolerance

REORGANIZE DBD=dbdname, SELECT_UOW=ALL, ERROR_THRESHOLD=3

Analyzing the DEDB during reorganization

If your site has a license for the Fast Path Analyzer/EP product, pointer validation will occur automatically (by default) when you execute an offline reorganization.

This automatic process applies the default value of QUICK for the POINTER_VALIDATION keyword. The analysis function performs a checksum validation of pointers for each segment type within each UOW of the database that is specified on the REORGANIZE command. Automatic pointer validation provides assurance of the area’s pointer integrity, while providing statistics that show how the reorganization process affected space usage.

NOTE

You can override automatic quick pointer validation and request more detailed validation by specifying POINTER_VALIDATION=FULL on the REORGANIZE or GLOBAL command. Direct pointers are validated by using the cross-reference technique. Although not recommended, you can also specify another valid value for the POINTER_VALIDATION keyword (such as POINTER_VALIDATION=NONE to completely disable the analysis function).

If an SDEP segment is defined for the database and the number of UOWs for IOVF control intervals is being extended (using the EXTEND_IOVF_#UOWS keyword), SDEP pointers are also validated (by default) by using the same technique as specified for direct pointers. You can manually specify any valid value for the SDEP_VALIDATION keyword to control how SDEP pointers are to be validated.

You can also specify additional functions that are associated with the analysis function:

■ Use the RAP_VALIDATION keyword to control how RAPs are processed.

■ Use the LARGEST_DATABASE_RECORDS keyword to specify the number of largest database records to be tracked by the analysis process.

■ Use the REPORT and THRESHOLD subcommands to control the generation of analysis reports and exception testing.
Making an image copy during reorganization

You can request that one or more output image copy data sets be produced during reorganization. Specify the IC subcommand for each image copy to be produced. For details about the IC subcommand, see Chapter 10, “Making an image copy.”

The example in Figure 43 requests the creation of a single output image copy data set for the area being reorganized.

Sample offline REORGANIZE command scenarios

For scenarios that show how to use the REORGANIZE command with key related keywords and subcommands, see Appendix C, “Sample command scenarios.”
Analyzing a DEDB

This chapter provides information on the capabilities, setup, and use of the offline analysis function, which is provided by the Fast Path Analyzer/EP product. This function lets you analyze and validate the content and structure of DEDBs, and lets you generate a variety of useful analysis reports. This chapter discusses the following topics:

Analysis function overview ................................................................. 88
Offline analysis .................................................................................. 90
  Inputs and outputs ............................................................................ 90
  Control statement ............................................................................ 91
  DBRC considerations ....................................................................... 91
  MADS considerations ..................................................................... 91
Analysis of an image copy ................................................................. 92
  Inputs and outputs ............................................................................ 92
  Control statement ............................................................................ 93
ANALYZE command keywords and subcommands ......................... 93
Selecting the Database and Areas ..................................................... 94
Allocating the area data set ................................................................. 95
Verifying area integrity ....................................................................... 96
  Physical validation ........................................................................... 96
  Pointer validation ........................................................................... 97
  RAP validation ................................................................................ 98
  SDEP validation .............................................................................. 99
Correcting pointer errors ................................................................... 100
  Restrictions for correcting pointer errors ........................................ 101
  Allocating the pointer error corrections data set ............................. 102
  Processing the generated output ...................................................... 103
Generating analysis reports ............................................................... 104
  Report selection ............................................................................... 104
  UOW range report specification ...................................................... 106
  Report formatting ............................................................................ 107
  Report routing ................................................................................ 107
Specifying the number of largest database records ........................... 108
Detecting exception conditions ........................................................ 109
Generating input to the Area Change Modeling Utility ...................... 110
Suppressing repetitious messages .................................................... 111
The Fast Path Analyzer/EP analysis function gathers the data required for assessing data entry database (DEDB) capacity and performance characteristics and for making DEDB maintenance decisions. Using the ANALYZE command, subcommands, and keyword parameters, you can perform the following tasks:

- analyze information about single or multiple areas
- validate pointer data
- generate reports
- set thresholds for exception condition detection
- retain data in a statistics repository
- retrieve the data for historical reporting
- generate input to the Area Change Modeling Utility feature
The analysis function analyzes and validates the content and structure of DEDBs. The function can analyze an area data set or an image copy of an area data set. The analysis function includes the following features:

- **user-specified keywords**

  Keywords and keyword parameters let you control and customize the analysis.

- **pointer validation**

  The validation function examines and validates pointer data during every analysis.

- **reporting**

  The analysis function computes statistics and analyzes information about single or multiple areas and presents the results in reports.

- **thresholds**

  User-specified values set threshold criteria by which to perform analysis to detect exception conditions.

- **statistics repository**

  The statistics repository can store information for historical reporting.

Fast Path Analyzer/EP includes the following utilities that can use input from the analysis function to improve database performance and integrity:

- **Area Change Modeling Utility**

  The Area Change Modeling utility lets you model a potential DEDB change before performing it.

- **SDEP Space Utilization Utility**

  The SDEP Space Utilization Utility lets you track usage of the SDEP portions of your area.

- **Control Interval Dump and Modification Utility**

  The command-driven Control Interval Dump and Modification Utility lets you dump, inspect, verify, or modify the entire contents of a database control interval (CI). This utility can be used to repair pointer errors and VSAM control information even when standard VSAM utilities do not provide access to the data. This utility can also be used to print the contents of the DMAC block for an area.
For more information about these utilities, see the *Fast Path/EP Series Reference Manual*.

## Offline analysis

Fast Path Analyzer/EP can read and analyze an offline DEDB.

Offline analysis operating characteristics are as follows:

- The DEDB is offline to IMS when analysis occurs.
- Read-with-integrity access is used.
- No other application can update the DEDB while analysis is in progress.

To analyze an offline DEDB, the ACCESS keyword is set to OFFLINE (the default value). During analysis, no other applications can update the database.

### Inputs and outputs

*Figure 44* shows the inputs and outputs for the offline analysis function.

*Figure 44  Offline Analysis Function*
Control statement

A sample control statement for an offline analysis is shown in Figure 45. In this example, the area data set and ACB library are accessed by using dynamic allocation. Analysis will be limited only to the area that is specified with the IAREA keyword.

Figure 45  Example JCL for an offline analysis

```
//PFP      EXEC PGM=PFPMAIN,REGION=0M
//STEPLIB  DD   DSN=BMC.PFP.LOAD,DISP=SHR
//         DD   DSN=IMS.RESLIB,DISP=SHR
//PFPSYSIN DD   *
ANALYZE DBD=dbname,IAREA=areaname
/*
```

DBRC considerations

When DBRC is active during execution of the offline analysis function, and the area is registered, a read-with-integrity authorization level is requested for the area data set. If the area data set is used as input, the data set name must match the name that is registered with DBRC.

MADS considerations

Multiple area data sets (MADS) are not supported by the offline analysis process. The product searches the area data set (ADS) list that is registered for each area (in collating sequence by DD name). The product selects the first ADS that is marked as available for use and that has no error queue elements (EQEs). If an ADS meets both of these criteria, it is the only ADS that is analyzed. All other area data sets are ignored.

If no area data set is marked as available, or if all available area data sets contain one or more EQEs, the analysis function is not performed for that area.
Analysis of an image copy

The analysis function can read and analyze an image copy of a DEDB. The operating characteristics during analysis of an image copy data set are as follows:

- The DEDB area data set is not accessed.
- Any standard image copy data set can be analyzed, including data sets that are created by using concurrent processing.

The characteristics of the analysis depend on the type of image copy being analyzed. Analysis of a batch image copy (one that represents a valid recovery point) resembles analysis of a DEDB by using the offline method. Analysis of a concurrent image copy (one that does not represent a recovery point) resembles analysis of a DEDB by using the concurrent method.

**NOTE**

When an offline analysis is performed with an image copy as input, no data will be sent to the Fast Path repository that is in MAXM Database Advisor for IMS, because the data set name of the image copy will not match the data set name of the area data set that is registered in MAXM Database Advisor.

**Inputs and outputs**

Figure 46 shows the analysis function with an image copy that is used as input.

**Figure 46  Analysis function using an image copy**

![Diagram of analysis function using an image copy]

- **PFPSYSIN** control statements
- **Repository**
  - validation information
  - threshold warnings
- **image copy**
- **ANALYZE command**
- **reports**
Control statement

The JCL for analysis of an image copy is shown in Figure 47. In this example, the input image copy and ACB library are accessed by using dynamic allocation.

**Figure 47 JCL example for an image copy analysis**

```plaintext
//PFP EXEC PGM=PFPMAIN,REGION=0M
//STEPLIB DD DSN=BMC.PFP.LOAD, DISP=SHR
//      DD DSN=IMS.RESLIB,DISP=SHR
//PFPPRINT DD SYSOUT=*  
//PFPSYSIN DD *
ANALYZE DBD=dbdname, IAREA=areaname, INPUT_DSN_MASK='dataset-name-mask'
```

ANALYZE command keywords and subcommands

Table 6 lists the keywords and subcommands that are available for the ANALYZE command.

**Table 6 ANALYZE command keywords and subcommands (part 1 of 2)**

<table>
<thead>
<tr>
<th>Function</th>
<th>Command or subcommand</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>selecting the database and areas</td>
<td>ANALYZE</td>
<td>DBD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IAREA</td>
</tr>
<tr>
<td>allocating the area data set</td>
<td>ANALYZE</td>
<td>INPUT_DSN_MASK</td>
</tr>
<tr>
<td>setting the operating mode</td>
<td>ANALYZE</td>
<td>ACCESS</td>
</tr>
<tr>
<td>verifying area integrity</td>
<td>ANALYZE</td>
<td>POINTER_VALIDATION</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RAP_VALIDATION</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDEP_VALIDATION</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ORPHANED_SDEP_MSG</td>
</tr>
<tr>
<td>correcting pointer errors</td>
<td>CORRECTIONS_FILECTL</td>
<td>DDNAME</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DSNAME</td>
</tr>
<tr>
<td>specifying the number of largest database</td>
<td>ANALYZE</td>
<td>LARGEST_DATABASE_RECORDS</td>
</tr>
<tr>
<td>records to be tracked during analysis</td>
<td></td>
<td></td>
</tr>
<tr>
<td>generating analysis reports</td>
<td>REPORT</td>
<td>all associated keywords</td>
</tr>
<tr>
<td>detecting exception conditions</td>
<td>THRESHOLD</td>
<td>all associated keywords</td>
</tr>
<tr>
<td>generating input for the Area Change</td>
<td>ANALYZE</td>
<td>MODEL_DDNAME</td>
</tr>
<tr>
<td>Modeling utility</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Selecting the Database and Areas

Table 6  ANALYZE command keywords and subcommands (part 2 of 2)

<table>
<thead>
<tr>
<th>Function</th>
<th>Command or subcommand</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>suppressing repetitious messages</td>
<td>ANALYZE</td>
<td>MESSAGE_SUPPRESSION</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ORPHANED_SDEP_MSG</td>
</tr>
<tr>
<td>using a history file</td>
<td>ANALYZE</td>
<td>HISTORY_DDNAME</td>
</tr>
<tr>
<td>enhancing performance</td>
<td>ANALYZE</td>
<td>ICACHE</td>
</tr>
<tr>
<td>producing an image copy during</td>
<td>IC</td>
<td>all associated keywords</td>
</tr>
<tr>
<td>analysis</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For more information about the following topics, see the *Fast Path/EP Series Reference Manual*:

- syntax of commands, subcommands, and keywords that are discussed in this book
- diagrams that show the syntax and available parameters and values for Fast Path Offline Suite commands and subcommands

**Selecting the Database and Areas**

The DBD keyword identifies the name of the DEDEB (DBD name) to be analyzed. The DBD keyword is required for offline analysis processing.

The IAREA keyword can be used to select specific areas to be analyzed. If you omit the IAREA keyword, all areas that are defined in the DEDEB are analyzed.

Areas can be specified on the IAREA keyword by using any combination of area names, area numbers, or a area ranges. The following parameters are available for the IAREA keyword:

- IAREA=ALL (default) or IAREA=* specifies all areas of the DEDEB.
- IAREA=areaname specifies one or more areas by using the one-character to eight-character area name for each area specified. Multiple area names must be enclosed in parentheses and separated by commas.
- IAREA=areanumber specifies one or more areas by using the one-character to five-character area number for each area specified. Multiple area numbers must be enclosed in parentheses and separated by commas.
- IAREA=(RANGE=(startarea,endarea)) specifies a consecutive range of areas using either areaname or areanumber parameters. The area number associated with startarea must be less than the area number associated with endarea.
An asterisk (*) can be used to specify all areas of the DEDB. When the * character is used with the RANGE keyword, it can be used to specify the beginning or ending range for specific areas of the DEDB.

To analyze an entire DEDB, use a command set like the example shown in Figure 48.

**Figure 48  Sample control statement for analyzing all areas**

```plaintext
ANALYZE DBD=dbname,AREA=ALL
```

To analyze specific areas, use a command set like the example shown in Figure 49.

**Figure 49  Sample control statement for analyzing specific areas**

```plaintext
ANALYZE DBD=dbname,AREA=(AREANAM1,AREANAM3,RANGE=(5,8))
```

This control statement uses a combination of area names and area numbers to request that the specified areas are to be analyzed. The areas named AREANAM1 and AREANAM3 will be analyzed. Also, all consecutive areas from area number 5 to area number 8 (area5, area6, area7 and area8) will be analyzed.

### Allocating the area data set

For offline analysis, or when analyzing an image copy, the area data set can be supplied in the JCL or can be accessed by using dynamic allocation.

The areaname DD statement identifies the area data set to be analyzed. The areaname DD statement can (as an option) specify an image copy data set. If the areaname DD statement is omitted from the JCL, Fast Path Reorg/EP attempts to dynamically allocate it.

The IMSACB DD statement identifies the ACB library containing the database definition that describes the area that is referenced by the areaname DD statement.

If you are using dynamic allocation, do not include the areaname DD statement. Fast Path Analyzer/EP tries to obtain the data set name for allocation in the following order:

1. If the INPUT_DSN_MASK keyword is specified, it is used to generate the data set name. The data set name can specify an image copy.

2. If DBRC is active and the area is registered, the registered area data set name is obtained from DBRC.

3. The STEPLIB is searched for the DFSMDA member that contains the data set name for this area.
If DBRC is active, the area is registered with DBRC, and the data set to be analyzed is *not* an image copy, the allocated data set name must match the registered data set name, regardless of how it is allocated.

Verifying area integrity

The analysis function validates the physical and logical integrity of the area. Physical verification is the process of validating that the area can be read and interpreted correctly. Logical verification is the process of validating that the pointer chains among segment types are correct.

Physical validation

Each time you execute an analysis process, the physical integrity of the area is verified. Physical validation performs the following tasks:

- Validates that the CIs are of the proper type and that the CIs are in the proper sequence
- Validates the IMS overhead bytes in the CI prefix, and validates the VSAM overhead bytes in the CI suffix
- Validates the free space element (FSE) chains
- Validates that each RAP, DOVF, IOVF, and SDEP CI is filled completely with segments, FSEs, and scraps
- Validates each allocation table entry and the free block chain within each Space Map CI
- Validates DOVF free block chains, and cross-validates each IOVF CI with the corresponding entry in the Space Map CIs
- Checks that each pointer to a root or direct dependent segment references a location within the direct portion of the area
- Checks that PCL and SSPTR pointers are null if the corresponding PCF pointer is null
- Checks that the PCL pointer is not null if the corresponding PCF pointer is not null
- Checks that each pointer to a sequential dependent segment references a location within the sequential portion of the area
Pointer validation

The analysis function validates the logical integrity of pointers to the root and direct dependent segments. You can specify an implicit value for the POINTER_VALIDATION keyword in the GLOBAL command. This keyword value is used by all analysis functions within the job step. To override the GLOBAL command value, specify the POINTER_VALIDATION keyword explicitly on the ANALYZE command.

- POINTER_VALIDATION=FULL constructs a complete cross-reference of every pointer and segment occurrence. When you specify POINTER_VALIDATION=FULL, the analysis process performs the following tasks:
  - checks every pointer to ensure that a segment of the proper type actually appears at the referenced relative byte address (RBA)
  - checks that every segment is referenced by one and only one physical child first (PCF) or physical twin forward (PTF) pointer
  - identifies the complete chain of dependent segment occurrences associated with their parent segment (the PCF/PTF pointers)
  - checks physical child last (PCL) and subset (SSPTR) pointers to verify that the segments which are referenced appear in the chain that is defined by the PCF/PTF pointers
  - checks all keyed segments for correct key sequencing to ensure that the sequence keys of logically adjacent segments are ascending and unique

**NOTE**

Because many statistics that are required for report generation can be collected only when POINTER_VALIDATION=FULL, certain reports are not produced if other values are selected. The following reports will not be generated if POINTER_VALIDATION=QUICK or POINTER_VALIDATION=OFF is selected:

- Segment I/O Analysis Report
- Record Length Analysis Report
- Record Placement Analysis Report
RAP validation

- **POINTER_VALIDATION=QUICK** (the default) generates a checksum for each segment type within each unit of work (UOW). The checksum consists of the sum of the RBAs for each segment occurrence minus the RBAs of all PCF/PTF pointers to the segment type. Because there should be exactly one PCF/PTF pointer to each segment, a non-zero value indicates a pointer problem. A second checksum consists of the sum of the RBAs for each segment with a null PTF pointer minus the RBAs of all PCL pointers to the segment type. A non-zero value indicates a problem. SSPTR pointers are not validated in this mode.

- **POINTER_VALIDATION=OFF** bypasses logical integrity verification of pointers to the root and direct dependent segments. The analysis function still reads the area to perform physical validation and to gather statistics.

- **POINTER_VALIDATION=NONE** specifies that the analysis function is not to be executed. This value is valid only on GLOBAL, REORGANIZE, RELOAD, CHANGE, and IMAGECOPY commands; it is not valid when used with the ANALYZE command. This value is used to completely cancel the analysis process as an override of the **POINTER_VALIDATION=QUICK** default, which otherwise occurs during these processes.

**RAP validation**

The method that is used to verify the logical integrity of root anchor point (RAP) and root segment pointers can be controlled independently by using the RAP_VALIDATION keyword. You can specify an implicit value for the RAP_VALIDATION keyword on the GLOBAL command. This keyword value is used by all analysis functions within the job step. To override the GLOBAL command value, explicitly specify the RAP_VALIDATION keyword on the ANALYZE command.

- **RAP_VALIDATION=(NOXREF,)** (the default) validates RAP and root segment pointers in the same manner as that specified by the **POINTER_VALIDATION** keyword.

- **RAP_VALIDATION=(XREF,)** constructs a complete cross-reference of every RAP and root segment occurrence. The analysis function checks every pointer to ensure that a root segment appears at the referenced location. The complete chain of root segments associated with the RAP is identified. The analysis function checks that every root segment occurrence is referenced by one (and only one) pointer.

- With **RAP_VALIDATION=(,NOPLACEMENT)** (the default), the placement of root segments is not validated.
With RAP_VALIDATION=(PLACEMENT), the placement of every root segment is validated by calling the randomizer (if available) to confirm the following placement:

— The root segment is located in the proper area.
— The root segment is located in the proper RAP.

### SDEP validation

The logical integrity of SDEP pointers can be verified when you execute the analysis function. You can specify an implicit value for the SDEP_VALIDATION keyword on the GLOBAL command. This keyword value is used by all analysis functions within the job step. To override the GLOBAL command value, explicitly specify the SDEP_VALIDATION keyword on the ANALYZE command. Any of the following modes of pointer validation can be selected by using the SDEP_VALIDATION keyword:

- **SDEP_VALIDATION=FULL** constructs a complete cross-reference of every pointer and segment occurrence. The analysis function checks every pointer to ensure that a segment occurrence appears at the referenced location. The complete chain of sequential dependents that are associated with the root segment are identified. The analysis function verifies that each segment occurrence is referenced by one (and only one) pointer.

- **SDEP_VALIDATION=QUICK** (the default) generates a checksum for all sequential dependents within the area. The checksum consists of the sum of the RBAs for each segment occurrence minus the RBAs of all SDEP pointers. Because there should be exactly one pointer to each segment, a non-zero value indicates a pointer problem.

- **SDEP_VALIDATION=OFF** bypasses logical verification of sequential dependent pointers. The product still reads the SDEP control intervals to perform physical validation and to gather statistics.

- **SDEP_VALIDATION=None** bypasses all processing of the SDEP control intervals. The product does not perform physical validation and does not gather statistics.

### Active SDEP boundaries

The boundaries of the active sequential dependent portion of the area are defined by the logical begin (LB) and logical end (LE) values that are stored within the DMAC. The interpretation of these values depends on the type of analysis being executed.
Correcting pointer errors

During offline analysis (or during analysis of a batch image copy), no applications are updating the database. Because of this, the LB and LE values stored in the DMAC exactly match the active boundaries of the SDEP portion. The analysis function scans SDEP segments only if they fall within these boundaries. If an 8-byte SDEP pointer is less than LB, it is treated as a null pointer (ignored). If an 8-byte SDEP pointer is greater than LE, it is reported as an error.

Orphaned sequential dependents

If a root segment is deleted, its sequential dependent segments become “orphans.” The segments still exist within the active sequential dependent portion of the area, but they are not referenced by a valid SDEP pointer. The SDEP validation modes handle orphaned sequential dependents as follows:

- SDEP_VALIDATION=NONE and SDEP_VALIDATION=OFF do not detect orphaned segments and will not report them as errors.
- SDEP_VALIDATION=QUICK reports a pointer error if an orphaned segment exists. This mode of pointer validation cannot distinguish between orphaned segments and other problems, and it is not recommended if orphaned segments are known to exist in the area.
- SDEP_VALIDATION=FULL identifies the individual segments as orphans.

You can use the ORPHANED_SDEP_MSG keyword to control the severity that is associated with this condition and the generation of messages.

NOTE

If you are running IMS version 6.1 or later, any SDEP with a timestamp less than the logical begin timestamp is treated as deleted. SDEP_VALIDATION=QUICK might report a pointer problem after SDEP scan/delete has been run. This condition occurs if a root segment is deleted, and that root segment was previously the owner of one of the logically deleted SDEP segments.

Correcting pointer errors

If pointer errors occur during the analysis process, you can use the optional CORRECTIONS_FILECTL subcommand with the ANALYZE command to correct pointer errors. If the CORRECTIONS_FILECTL subcommand is specified, pointer corrections control cards are created and then written to a pointer corrections output data set that can be used to zap pointer errors to zero. If the CORRECTIONS_FILECTL subcommand is not specified, no pointer corrections control cards are created, and invalid pointers are not removed.
Restrictions for correcting pointer errors

Pointer corrections control cards can only be generated for direct pointers or sequential pointers. As a result, POINTER_VALIDATION=FULL or SDEP_VALIDATION=FULL must also be specified with the ANALYZE command and the CORRECTIONS_FILECTL subcommand to correct pointer errors.

The following restrictions apply to correcting errors for direct pointers or sequential pointers:

- To generate pointer corrections records for direct pointers, POINTER_VALIDATION=FULL must be specified with the Analyze function.

- To generate pointer corrections records for sequential pointers, SDEP_VALIDATION=FULL must be specified with the Analyze function.

**NOTE**

If POINTER_VALIDATION=FULL is specified, SDEP_VALIDATION defaults to FULL.

- Pointer corrections control cards can only be generated for the following errors:

<table>
<thead>
<tr>
<th>Message number</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC121190E</td>
<td>dbdname, areaname, rba: Segment=segnamel segtype to segment=segname2 - pointer=value out of bounds</td>
</tr>
<tr>
<td>BMC121191E</td>
<td>dbdname, areaname, rba: Segment=segnamel segtype to segment=segname2 - pointer=(cycle:rba) out of logical bounds</td>
</tr>
<tr>
<td>BMC121192E</td>
<td>dbdname, areaname, rba: Segment=segnamel segtype to segment=segname2 - SDEP pointer=(cycle:rba) out of physical bounds</td>
</tr>
<tr>
<td>BMC121193E</td>
<td>dbdname, areaname, rba: Segment=segnamel ptrtype to segment=segname2 - segment points to itself</td>
</tr>
<tr>
<td>BMC121194E</td>
<td>dbdname, areaname, rba: Segment=segnamel ptrtype to segment=segname2 PCF pointer=00000000, ptrtype pointer=value</td>
</tr>
<tr>
<td>BMC121200E</td>
<td>dbdname, areaname, rba: Segment=segnamel found - segment=segname2 expected Segment=segname3 (at rba2) contains a ptrtype pointer to segment=segname2 (at rba1)</td>
</tr>
<tr>
<td>BMC121204E</td>
<td>dbdname, areaname, rba1: Segment=segnamel found - segment=segname2 expected RAP (at rba2) points to root segment=segname2 (at rba1)</td>
</tr>
</tbody>
</table>
Allocating the pointer error corrections data set

The pointer error corrections output data set can be allocated by using JCL DD statements or by using dynamic allocation. When the CORRECTIONS_FILECTL subcommand is specified, along with POINTER_VALIDATION=FULL or SDEP_VALIDATION=FULL on the ANALYZE command, a pointer error corrections output data set is always written. However, if no pointer errors are encountered, the pointer error corrections output data set will be empty. By default, only one output pointer error corrections data set is produced for each input area; you cannot combine multiple areas into a single data set.

**Using CORRECTIONS_FILECTL with standard JCL**

Use the DDNAME keyword (*not* the DSNAME keyword) to indicate that the pointer error corrections data set is to be allocated by using JCL DD statements.

**NOTE**

When using JCL to allocate the pointer error corrections data set, ensure that only one area is written to any DD statement. If more than one area is written to the same DD statement, the pointer error corrections data set of the second area will overwrite the pointer error corrections data set of the first area.

The example in Figure 50 uses JCL to allocate a single pointer error corrections data set for an area.

**Figure 50  JCL to allocate a pointer error corrections data set by using the DDNAME keyword**

```bash
//PFP EXEC PGM=DFSRRC00,REGION=0M,
   //       PARM=(IFP,dbdname,DBF#FPUO)
//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
//         DD DISP=SHR,DSN=IMS.RESLIB
//ddname DD UNIT=TAPE,DISP=(NEW,CATLG),
//         DSN=your.dataset.name
//PFPSYSIN DD *
ANALYZE DBD=dbdname,AREA=areaname,
   //         POINTER_VALIDATION=FULL,
   //         CORRECTIONS_FILECTL DDNAME=ddname
/*
Using CORRECTIONS_FILECTL with dynamic allocation

Use the DSNAME keyword (*not* the DDNAME keyword) to indicate that the pointer error corrections data is to be dynamically allocated and to specify the name of the data set where the error corrections control cards are to be written. Use the DSNAME keyword to supply a mask to construct the data set name. Substitution variables within the mask can be used to generate unique names for each output data set.

Additional related keywords, such as UNIT, DISP, and DATAACLAS, can be used to control the allocation and disposition of the data set, similar to the corresponding JCL keywords.

The example in Figure 51 shows how to generate a pointer error corrections data set by using dynamic allocation.

**Figure 51 Dynamic allocation of pointer error corrections data set by using DSNAME keyword**

```plaintext
ANALYZE DBD=dbdname
    POINTER_VALIDATION=FULL,
    CORRECTIONS_FILECTL,
    DSNAME='dataset-name-mask',
    UNIT=TAPE,DISP=(NEW,CATLG)
```

Processing the generated output

The output that is written to the pointer error corrections data set can be used to correct pointer errors. The PROCESS_AREA command and the PERFORM subcommand are automatically generated by the analysis process (and placed in the pointer error corrections data set) for each area that contains one or more pointer errors. In addition, a set of VER, REP, and COMMIT parameters is also generated for each pointer error.

The example in Figure 52 shows how to process the output that is written to the pointer error corrections data set.

**Figure 52 JCL to process the generated output**

```plaintext
//PROCESS EXEC PGM=DFSRRC00,REGION=0M,
    //      PARM=(IFP,dbname,DFB#FPUO)
//areaname DD DISP=OLD,DSN=areadatasetname
//PFPPRINT DD SYSOUT=* 
//PFPRPTS DD SYSOUT=* 
//PFPSYSIN DD DISP=SHR,DSN=correctionsfilename
/*
Generating analysis reports

The REPORT subcommand controls the generation of analysis function reports. The REPORT subcommand lets you specify characteristics of output reports. This subcommand works whether analysis is invoked directly by using the ANALYZE command or by another BMC utility that uses the appropriate command or keyword.

The REPORT subcommand defines a report set which consists of analysis selections, UOW range selection, report formatting options, and routing options. Multiple REPORT subcommands can be used to request the generation of multiple report sets. Each report set is independent of every other report set.

You can specify reporting controls by placing your REPORT subcommands within the GLOBAL command. These reporting specifications are used by all analysis functions within the job step. To override these reporting specifications, place one or more REPORT subcommands within the ANALYZE command.

Report selection

The analysis function can produce any or all of 11 reports that fall into three categories. These reports contain detailed data to help you understand every aspect of database performance, space usage, segment characteristics, and record characteristics. As shown in Table 7, a keyword is available to request each report individually.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Report Title</th>
<th>Report Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>space usage</td>
<td>Free Space Analysis</td>
<td>FREESPACE_ANALYSIS</td>
</tr>
<tr>
<td></td>
<td>IOVF Space Analysis</td>
<td>IOVF_SPACE_ANALYSIS</td>
</tr>
<tr>
<td></td>
<td>UOW Detailed Analysis</td>
<td>UOW_DETAILED_ANALYSIS</td>
</tr>
<tr>
<td>segment</td>
<td>Pointer Analysis</td>
<td>POINTER_ANALYSIS</td>
</tr>
<tr>
<td></td>
<td>Segment I/O Analysis</td>
<td>SEGMENT_IO_ANALYSIS</td>
</tr>
<tr>
<td></td>
<td>Segment Length Analysis</td>
<td>SEGMENT_LENGTH_ANALYSIS</td>
</tr>
<tr>
<td></td>
<td>Segment Placement Analysis</td>
<td>SEGMENT_PLACEMENT_ANALYSIS</td>
</tr>
<tr>
<td>record</td>
<td>Record Length Analysis</td>
<td>RECORD_LENGTH_ANALYSIS</td>
</tr>
<tr>
<td></td>
<td>Record Placement Analysis</td>
<td>RECORD_PLACEMENT_ANALYSIS</td>
</tr>
<tr>
<td></td>
<td>Record Profile Analysis</td>
<td>RECORD_PROFILE_ANALYSIS</td>
</tr>
<tr>
<td></td>
<td>Synonym Chain Analysis</td>
<td>SYNONYM_CHAIN_ANALYSIS</td>
</tr>
</tbody>
</table>
Two of these reports use additional customization keywords:

- The SYNONYM_CHAIN_INCREMENT keyword specifies the reporting interval for the Synonym Chain Analysis Report. This keyword is specified on the REPORT subcommand.

- The RECORD_LENGTH_INCREMENT keyword specifies the record length interval for the Record Length Analysis Report. This keyword is specified on the REPORT subcommand.

- The LARGEST_DATABASE_RECORDS keyword specifies the number of largest database records to be reported on the Record Length Analysis Report. This keyword can be specified under any primary command that enables you to request analysis with the POINTER_VALIDATION keyword.

**NOTE**

You must specify POINTER_VALIDATION=FULL on the primary command to generate the Record Length Analysis Report. The LARGEST_DATABASE_RECORDS keyword implements the tracking of these records only; it does not generate the report. For more information about the function of the LARGEST_DATABASE_RECORDS keyword, see page 108.

For summary descriptions of these reports, see “Analysis report descriptions” on page 118. For more detailed descriptions of these reports, see the Fast Path/EP Series Reference Manual. For examples of the reports, see the PFPANLYZ member in the REPORTS data set.

### Requesting all reports

You can use the REPORT_DEFAULT keyword to specify whether all reports are produced by default. If you specify REPORT_DEFAULT=YES, all reports are produced unless you suppress individual reports by using the associated keywords.

### Excluding selected reports

The sample command set in Figure 53 specifies that analysis will issue all reports except for the UOW Detailed Analysis Report.

**Figure 53  Sample JCL specifying special reporting to exclude one report**

```
ANALYZE DBD=dbdname, IAREA=areaname
REPORT
  REPORT_DEFAULT=YES,
  UOW_DETAILED_ANALYSIS=NO
```
Producing selected reports

If you specify REPORT_DEFAULT=NO, no reports are produced unless you select individual reports by using the associated keywords. The sample command set in Figure 54 specifies that only the Free Space Analysis Report be produced.

Figure 54 Sample JCL specifying special reporting to include one report

```
ANALYZE DBD=dbname,IAREA=areaname
REPORT
  REPORT_DEFAULT=NO,
  FREESPACE_ANALYSIS=YES
```

Unless POINTER_VALIDATION=FULL is specified, the information that is required for generation of certain reports is not collected. The following reports are generated only in FULL validation mode:

- Segment I/O Analysis
- Record Length Analysis
- Record Placement Analysis

The information to generate the Synonym Chain Analysis Report is collected if POINTER_VALIDATION=FULL or RAP_VALIDATION=XREF is specified. If neither of these keyword values is specified, the report is not generated.

UOW range report specification

Keywords that specify the UOW range on which to report include the following:

- STARTUOW specifies the UOW on which to start reporting.
- STOPUOW specifies the UOW on which to stop reporting.

You can use the STARTUOW and STOPUOW keywords to specify a range of UOWs to be included in the report set.

To limit the range of UOWs to be analyzed, use the command set in Figure 55.

Figure 55 Sample JCL limiting the range of UOWs to be analyzed

```
ANALYZE DBD=dbname,IAREA=areaname
REPORT
  STARTUOW=startuow,
  STOPUOW=stopuow
```
The report set includes database \textit{dbdname}, area \textit{areaname}, starting with UOW \textit{startuow} and ending with UOW \textit{stopuow}. These keywords have no effect on the verification processing; the complete area is always verified. You can use the keywords to limit reporting to certain UOWs. Some reports are always area-wide, regardless of UOW range selection.

If the HISTORY_DDNAME keyword has been specified, a separate history record is produced for each UOW range that is specified.

\section*{Report formatting}

You can use the REPORT\_HEADING keyword to specify a value to be generated in the page heading of each output page in the report set.

You can use the REPORT\_LINE\_COUNT keyword to specify the maximum number of printed lines to be placed on each output page within the report set.

\section*{Report routing}

The destination of reports can be designated by using the REPORT\_DDNAME keyword. You can use the JCL parameters on the referenced DD statements to place reports on tape, disk, or other storage medium.

You can generate multiple copies of a report by listing more than one ddname. You can also route more than one report set to the same destination by using the same ddname on multiple REPORT subcommands.

\textbf{WARNING}

Because the DD statement is opened and closed for each report set, you must use DISP=MOD in these cases.

The example in Figure 56 uses two REPORT subcommands. The first set of reports is written to a tape data set; the second REPORT subcommand requests the Free Space Analysis Report to be routed to the default destination.

\begin{figure}
\begin{lstlisting}
//PFP EXEC PGM=PFPMAIN,REGION=0M
//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
//                   DD DISP=SHR,DSN=IMS.RESLIB
//REPORTS DD UNIT=TAPE,DSN=your.dataset.name,
//          DISP=(NEW,CATLG)
//PFP SYSIN DD *
\end{lstlisting}
\caption{Sample JCL for special use of the REPORT subcommand (part 1 of 2)}
\end{figure}
Specifying the number of largest database records

Fast Path Analyzer/EP incorporates a method that allows you to specify the number of the largest database records that are tracked by the analysis process. By default, Fast Path Analyzer/EP retains information on the 10 largest database records when you specify full pointer analysis by using the POINTER_VALIDATION keyword.

You can request a different number of largest database records by specifying a desired value on the LARGEST_DATABASE_RECORDS keyword. This keyword can be specified under any primary command that enables you to request analysis with the POINTER_VALIDATION keyword. When you specify the LARGEST_DATABASE_RECORDS keyword, the process will reserve adequate storage to track the number of largest database records that you request.

If you have activated the Fast Path Analyzer/EP statistics repository facility, the information that is gathered by the analysis process is stored in the repository for future retrieval. The information on the largest database records can also be obtained at a later time by requesting the Record Length Analysis Report from the repository using any of the available methods. For more information, see Chapter 11, “Statistics repository facility.”

The example shown in Figure 57 will track information on the 30 largest database records contained in the specified area.

Figure 57  Specifying the LARGEST_DATABASE_RECORDS keyword

```
ANALYZE DBD=dbname,IAREA=areaname.
POINTER_VALIDATION=FULL,
LARGEST_DATABASE_RECORDS=30
```

The example shown in Figure 58 will also track information on the 30 largest database records contained in the area. The REPORT subcommand is specified on the command set to generate the Record Length Analysis Report. This report will make the information available on these largest database records.

Figure 56  Sample JCL for special use of the REPORT subcommand (part 2 of 2)

```
ANALYZE DBD=dbname,IAREA=areaname
REPORT
  REPORT_DDNAME=REPORTS
REPORT
  REPORT_DEFAULT=NO,
  FREESPACE_ANALYSIS=YES
/*
```
Detecting exception conditions

To facilitate the monitoring of particular database characteristics during analysis, the analysis function provides the THRESHOLD subcommand and its associated keywords to indicate database threshold settings. Using these keywords, you can specify that warning messages be sent to the processing log when certain database conditions are detected. Threshold tests are performed only when you specify the corresponding threshold keywords. Any threshold exception conditions that are detected are shown also on the Area Summary Report. The threshold keywords are categorized into the general groups that are listed in Table 8.

Table 8   Threshold types and keywords

<table>
<thead>
<tr>
<th>Threshold Type</th>
<th>Threshold Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>free space</td>
<td>DOVF_FREESPAC PERCENT</td>
</tr>
<tr>
<td></td>
<td>FREESPACE_DOVF_IOVF</td>
</tr>
<tr>
<td></td>
<td>FREESPACE_RAA_DOVF</td>
</tr>
<tr>
<td></td>
<td>FREESPACE_RAA_IOVF</td>
</tr>
<tr>
<td></td>
<td>IOVF_FREESPAC PERCENT</td>
</tr>
<tr>
<td></td>
<td>IOVF_USED PERCENT</td>
</tr>
<tr>
<td></td>
<td>RAA_FREESPAC PERCENT</td>
</tr>
<tr>
<td></td>
<td>RAP_OVERFLOW PERCENT</td>
</tr>
<tr>
<td>I/O</td>
<td>RECORD_IO_AVERAGE</td>
</tr>
<tr>
<td></td>
<td>RECORD_IO_MAXIMUM</td>
</tr>
<tr>
<td></td>
<td>ROOT_IO_AVERAGE</td>
</tr>
<tr>
<td></td>
<td>ROOT_IO_MAXIMUM</td>
</tr>
<tr>
<td>RAP usage</td>
<td>SYNONYM_CHAIN_LENGTH</td>
</tr>
<tr>
<td></td>
<td>SYNONYM_CHAIN_MAXIMUM</td>
</tr>
<tr>
<td></td>
<td>SYNONYM_CHAIN_PERCENT</td>
</tr>
<tr>
<td>overflow characteristic</td>
<td>RECORD_IOVF_PERCENT</td>
</tr>
<tr>
<td></td>
<td>UOW_DOVF_PERCENT</td>
</tr>
<tr>
<td></td>
<td>UOW_IOVF_AVERAGE</td>
</tr>
<tr>
<td></td>
<td>UOW_IOVF_MAXIMUM</td>
</tr>
<tr>
<td></td>
<td>UOW_IOVF_PERCENT</td>
</tr>
</tbody>
</table>
You can specify thresholds by placing the THRESHOLD subcommand within the GLOBAL command. These implicit threshold specifications are used by all analysis functions within the job step. To override the GLOBAL command, you can explicitly specify a THRESHOLD subcommand within the ANALYZE command. In the example shown in Figure 59, thresholds that are specified on the GLOBAL command are tested for all areas except area2 of database2. Specifying THRESHOLD without any keywords under the last ANALYZE command overrides the GLOBAL command, so no threshold checking is performed for area2.

Figure 59 Using the THRESHOLD subcommand with the GLOBAL command

```
//PFP EXEC PGM=PFPMAIN,REGION=0M
//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
// DD DISP=SHR,DSN=IMS.RESLIB
// IMSACB DD DISP=SHR,DSN=IMSVS.ACBLIB
GLOBAL
   REPORT
      RECORD_PROFILE_ANALYSIS=NO
   THRESHOLD
      FREESPACE_RAA_DOVF=(25,25),
      FREESPACE_RAA_IOVF=(50,50)
ANALYZE DBD=database1
ANALYZE DBD=database2,AREA=(area1,area3,area4)
ANALYZE DBD=database2,AREA=area2
THRESHOLD
/*
```

Generating input to the Area Change Modeling Utility

To generate input to the Area Change Modeling Utility, use the JCL for the analysis function and specify the following identifiers:

- Add DD statements identifying the segment information data set to be created.
- Use the MODEL_DDNAME keyword with the ANALYZE command to identify the names of the DD statements in the JCL.

NOTE
Record I/O thresholds and the RECORD_IOVF_PERCENT overflow threshold can be tested only if POINTER_VALIDATION=FULL has been requested. Root I/O thresholds and RAP usage thresholds can be tested if POINTER_VALIDATION=FULL or RAP_VALIDATION=XREF has been specified.
You can generate multiple copies of a report by listing more than one ddname.

The segment information data set is used as input to the Area Change Modeling Utility. Sample JCL is shown in Figure 60. For more information about the Area Change Modeling Utility, see the *Fast Path/EP Series Reference Manual*.

### Figure 60 Sample JCL for generating input to modeling utility

```plaintext
//PFPP EXEC PGM=PFPPMAIN,REGION=OM
//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
// DD DISP=SHR,DSN=IMS.RESLIB
//ddname1 DD DISP=OLD,DSN=your.dataset.name
//PFPSYSIN DD *
 ANLIZE DBD=dbdname,
 MODEL_DDNAME=ddname1
/*
```

### Suppressing repetitious messages

For each anomaly that is encountered in an area, the analysis function generates a message with a specific suffix (severity) level. When the number of places that a particular condition exists is large, a large number of messages is produced.

You can reduce the number of repetitious messages by using the MESSAGE_SUPPRESSION keyword. Use this keyword to specify the maximum number of times that any given informational, warning, error or critical message is to be produced. In the example in Figure 61, the MESSAGE_SUPPRESSION keyword is used to specify that a warning message be issued no more than 10 times and that an error message be produced no more than 15 times. When messages are suppressed in this way, Fast Path Analyzer/EP produces a summary of the number of times that each message has been suppressed.

### Figure 61 Using the MESSAGE_SUPPRESSION keyword

```plaintext
ANLIZE DBD=dbdname,AREA=areaname,
 MESSAGE_SUPPRESSION=(10,15)
```

To suppress the generation of orphaned SDEP error messages, you can specify ORPHANED_SDEP_MSG=NONE as shown in Figure 62.

### Figure 62 Using the ORPHANED_SDEP_MSG keyword

```plaintext
ANLIZE DBD=dbdname,
 ORPHANED_SDEP_MSG=NONE
```
Using a history file

Fast Path Analyzer/EP provides a facility to generate a file containing historical summary records. BMC recommends the use of the repository facility for storing historical data. This data can be collected based on the analysis of multiple areas, and can be generated in a single report. For more information, see Chapter 11, “Statistics repository facility.”

Each time an area is analyzed, a statistics record is appended to a user-specified file. The file can be used as input to a user-supplied reporting facility.

Use the HISTORY_DDNAME keyword on the ANALYZE command to specify the ddnames of DD statements to which the records are to be written. You can generate multiple copies of a report by listing more than one ddname.

---

**WARNING**

To prevent destructive concurrent access to the data set, the DD statement that is used for the output data should specify DISP=OLD as shown in Figure 63.

---

Use the sample JCL command set in Figure 63 to write statistics to the history file.

**Figure 63 Using the HISTORY_DDNAME keyword**

```
//PFP      EXEC PGM=PFPMAIN,REGION=0M
//STEPLIB  DD   DSN=BMC.PFP.LOAD,DISP=SHR
//        DD   DSN=IMS.RESLIB,DISP=SHR
//ddname1  DD DSN=your.dataset.name,DISP=OLD
//PFPSSIN DD *
  ANALYZE DBD=bdname,IAREA=areaname.
  HISTORY_DDNAME=ddname1
/*
```

The complete history file record layout is shown in the Fast Path/EP Series Reference Manual.

UOW range specification

An additional record is written to the history file for each UOW range that is requested on any REPORT subcommand for the area being processed. The record will contain statistics specific to the range of UOWs that are requested.

The sample JCL command set in Figure 64 will write two records to the history file. One record will contain statistics for the entire area; the other record will contain statistics only for the UOWs that are specified.
Enhancing performance

Using the input caching feature with offline analysis or image copy analysis can speed the process in many circumstances. Whether input caching is useful and significant to the analysis function depends on the type of input (image copy or area data set) and on the internal operating mode of the function.

The analysis function has two internal modes of operation: sequential mode and direct mode. Fast Path Analyzer/EP decides internally which mode is required, or which is more efficient, and executes accordingly. If the product performs the analysis function in sequential mode, any caching that is specified by using the ICACHE keyword is ignored.

Image copy input

Without caching, the analysis function must use sequential mode with image copy input. By using the ICACHE keyword to specify IOVF caching, the analysis function can use direct mode, performing random input I/O against the cached CIs. If resources are not available, the analysis function ignores any cache request, generates a message, and continues in sequential mode.

If resources allow, caching of IOVF and SDEP allows the analysis function to perform more efficiently. SDEP control intervals are not processed randomly, but loading them into the cache at the same time as the IOVF control intervals reduces I/O against the image copy data set.
Area data set input

If the analysis function is using an area data set as input, using the ICACHE keyword to specify IOVF input caching can significantly improve performance by eliminating the random I/O to the IOVF control intervals.

**NOTE**

Do not specify input caching of SDEP CIs. Because SDEP CIs are processed sequentially, caching provides no benefit and can slow the process.

Producing an image copy during analysis

You can request that one or more output image copy data sets be produced during analysis. Specify the IC subcommand for each image copy to be produced. The example in Figure 65 requests the creation of a single output image copy data set for the area being analyzed. For details about the IC subcommand, see Chapter 10, “Making an image copy.”

**Figure 65  Requesting an image copy during analysis**

```
ANALYZE DBD=dbdname,IAREA=areaname
    IC DSNAME='dataset-name-mask',UNIT=TAPE,DISP=(NEW,CATLG)
```

Image copy output is not supported when the input to the analysis function is an image copy data set.

Requesting analysis during reorganization

If your site has a license for Fast Path Analyzer/EP, it performs pointer validation automatically (by default) when you execute an offline reorganization by using the Fast Path Reorg/EP REORGANIZE command.

This automatic process applies the default value QUICK for the POINTER_VALIDATION keyword. The analysis function performs a checksum validation of pointers for each segment type within each UOW of the database that is specified on the REORGANIZE command.

Automatic pointer validation provides assurance of the area’s pointer integrity, while providing statistics that show how the reorganization process affected space usage.
If an SDEP segment is defined for the database and the number of UOWs for IOVF control intervals is being extended (using the EXTEND_IOVF/#UOWS keyword), SDEP pointers are also validated (by default) by using the same technique as specified for direct pointers. You can manually specify another valid value for the SDEP_VALIDATION keyword to control how SDEP pointers are to be validated.

You can also specify additional functions that are associated with the analysis function:

- Use the RAP_VALIDATION keyword to control how RAPs are processed.
- Use the LARGEST_DATABASE_RECORDS keyword to specify the number of largest database records to be tracked by the analysis process.
- Use the REPORT and THRESHOLD subcommands to control the generation of analysis reports and exception testing.

Because no values are specified for the POINTER_VALIDATION and SDEP_VALIDATION keywords, the example in Figure 66 requests that pointer validation be performed during reorganization of the area by using the QUICK (checksum) technique and that no validation of SDEP pointers is performed. Full validation for RAP pointers is requested explicitly. The example produces only the Free Space Analysis Report and does not check threshold conditions.

**Figure 66  Control statements for requesting analysis during reorganization**

```
REORGANIZE DBD=dbdname,
  RAP_VALIDATION=XREF,
  REPORT
  REPORT_DEFAULT=NO,
  FREESPACE_ANALYSIS=YES
```

For more information about the reorganize process, see Chapter 3, “Reorganizing a DEDB.”
Requesting analysis during DEDB change or reload

If your site has a license for Fast Path Analyzer/EP, it performs pointer validation automatically (by default) when you execute an offline database change or reload by using the CHANGE or RELOAD commands available in the Fast Path Reorg/EP product.

This automatic process applies the default value QUICK for the POINTER_VALIDATION keyword. The analysis function performs a checksum validation of pointers for each segment type within each UOW of the database that is specified on the CHANGE or RELOAD command.

Automatic pointer validation provides assurance of the area’s pointer integrity, while providing statistics that show how the database change or reload process affected space usage.

**NOTE**

You can override automatic quick pointer validation and request more detailed validation by specifying POINTER_VALIDATION=FULL on the CHANGE, RELOAD, or GLOBAL command. Direct pointers are validated by using the cross-reference technique. You can also specify any other valid value for the POINTER_VALIDATION keyword (such as POINTER_VALIDATION=NONE to completely disable the analysis process).

If an SDEP segment is defined for the database, SDEP pointers are also validated (by default) using the same technique as specified for direct pointers. You can manually specify another valid value for the SDEP_VALIDATION keyword to control how SDEP pointers are to be validated.

You can also specify additional functions that are associated with the analysis function. Use the RAP_VALIDATION keyword to control how RAPs are processed. Use the REPORT and THRESHOLD subcommands to control the generation of analysis reports and exception testing.

The example in Figure 67 requests that analysis be performed on all output areas created by the CHANGE command. Direct pointers are validated by specifying a value of FULL for the POINTER_VALIDATION keyword. SDEP pointers (if any exist) are validated by using the same technique. Default reports are produced, and no threshold checking is performed. For more information, see “Pointer validation” on page 97.

**Figure 67  Control statements for requesting analysis during DEDB change**

```
CHANGE DBD=dbname,
   POINTER_VALIDATION=FULL.
```
For more information about the change function, see Chapter 5, “Changing DEDB structure in place.” For more information about the reload function, see Chapter 6, “Unloading and Reloading a DEDB.”

Requesting analysis with IMAGE COPY PLUS or RECOVERY PLUS

You can invoke the Fast Path Analyzer/EP analysis function from within the BMC IMAGE COPY PLUS (ICP) or RECOVERY PLUS for IMS (RVP) products. This capability allows the analysis and utility functions to share the database I/O. You must have a license for Fast Path Analyzer/EP to perform this invocation and I/O sharing.

To request that the analysis function be invoked by these utilities, code the utility JCL and control statements normally. For detailed information about the JCL and control statements that are required, see the IMAGE COPY PLUS Reference Manual and the RECOVERY PLUS for IMS Reference Manual. You must code FPA(Y) on the appropriate utility control statements to request that the analysis function be invoked.

You must include the Fast Path/EP product library within the STEPLIB concatenation. Unless all libraries within the STEPLIB concatenation are APF authorized, the utility must have access to the BMC Software Database Utilities Subsystem.

The analysis function requires access to the ACB library containing the definitions for the areas to be analyzed. The IMSACB DD statement can be included in your JCL or accessed by using dynamic allocation. For more information about allocation of the ACBLIB, see “Dynamic allocation of data sets” on page 46.

The example JCL in Figure 68 shows how to invoke the analysis function from IMAGE COPY PLUS. The JCL to invoke the analysis function from RECOVERY PLUS is similar to this example.

Figure 68 Invoking the analysis function from IMAGE COPY PLUS (part 1 of 2)

```sql
//ICP   EXEC   PGM=ICPUMAIN,...
//STEPLIB DD ...
//       DD  DISP=SHR,DSN=BMC.PFP.LOAD
.
.
//ICPSYSIN DD *
IC  DBD(dbdbname)  AREA(AREANAME) FPA(Y)...
.
.
```
The analysis function generates commands internally to control the analysis process for the selected areas. The internally generated commands are shown in Figure 69.

You can include a PFPSYSIN DD statement to specify different ANALYZE keywords and subcommands. If you code an ANALYZE command with a matching DBD name and IAREA name, your command overrides the internally generated command. You can also use the GLOBAL command to override analysis parameters.

If your PFPSYSIN control statements contain commands other than GLOBAL and ANALYZE, these commands are ignored. Any ANALYZE commands that specify a database or area not being processed by the ICP or RVP utility are ignored.

The following keywords are ignored on any ANALYZE command:

- INPUT_DSN_MASK
- ICACHE
- ACCESS

The IC subcommand is ignored if it is coded under the ANALYZE or GLOBAL command.

You can include a PFPOPTS DD statement to specify any Fast Path Analyzer/EP options, such options that are required for requesting repository processing.

Analysis report descriptions

Reports can be produced during an offline analysis process. This section provides brief descriptions of these reports. For detailed descriptions of each report, see the Fast Path/EP Series Reference Manual.
Area Summary Report

The Area Summary Report provides basic information about each area analyzed. It is generated automatically with each report set and precedes all other reports. Threshold exception conditions are shown on the report.

Free Space Analysis Report

The Free Space Analysis Report provides the following information:

- summary and detailed information about the total and usable amount of free space in each section of the database (RAA base, DOVF, and IOVF)
- statistics about the number and size of the free space elements in each section of the database
- overflow usage analysis (DOVF and IOVF)

For an example of the report, see the PFPANLYZ member in the REPORTS data set.

The Free Space Analysis Report provides valuable information that can be used to determine the following items:

- when to reorganize or expand an area
- optimal values for UOW and ROOT parameters
- DOVF and IOVF usage

Free space calculations have the following characteristics:

- The entire area is included in the report. When an optional UOW range report is requested, the report also contains a summary of the free space within the selected UOWs.

- The total usable space in a CI is the block size minus 21 (overhead bytes).

- The length of the free space element (FSE) is included in usable free space only if it is large enough to hold the smallest maximum segment defined in the area.

- Scraps (free space less than 4 bytes) are included in free space calculations, but they are not included in FSE calculations.
IOVF Space Analysis Report

The IOVF Space Analysis Report provides IOVF block usage information that is obtained from each IOVF space map block to show how IOVF is being allocated. All space map blocks in the area are included in the report. The range scale indicates the amount of full CIs in the IOVF section.

UOW Detailed Analysis Report

The UOW Detailed Analysis Report provides detailed information about each UOW. You can use the report to perform the following tasks:

- identify which UOWs require reorganization
- manage space usage when the randomizer has been modified to group related data into contiguous UOWs

Pointer Analysis Report

For each segment type within a database, the Pointer Analysis Report provides detailed information for each of the following prefix pointer types:

- physical twin forward (PTF)
- physical child first (PCF) and physical child last (PCL)
- subset (SSPTR) and sequential dependent (SDEP) pointers

Additionally, each root anchor point (RAP) within the database is treated as a special segment type so that RAP chain pointers are included in this report. The following statistics are provided for each pointer type:

- count and percentage of null pointers
- count and percentage of pointers that point into the same CI
- count and percentage of pointers that point into a different CI

Segment I/O Analysis Report

The Segment I/O Analysis Report provides valuable information for database performance tuning and for evaluation of I/O requirements for specific user transactions. Analysis is used also to determine the optimal reorganization point and to validate the effects of UOW and ROOT value parameter changes.
The Segment I/O Analysis Report can be generated only when POINTER_VALIDATION=FULL is specified on the primary command.

The Segment I/O Analysis Report provides the following statistics for a typical database record:

- direct dependent segments in each part of an area (SDEPs segments are not included)
- physical I/O operations required to retrieve dependent segments
- physical I/O required to retrieve an average database record
- physical I/O required to retrieve an average root segment

**Segment Length Analysis Report**

The Segment Length Analysis Report shows the data lengths of segments in 21 reporting intervals (or less). In each interval, statistics are provided on the number of segments and their lengths. This report lets you see the level of compression for compressed segments.

**Segment Placement Analysis Report**

The Segment Placement Analysis Report provides an overview of where segments reside in the database. This report can provide information when to use the load control function in Fast Path Reorg/EP.

---

**NOTE**

SDEP segments are not included in this report.

---

**Record Length Analysis Report**

The Record Length Analysis Report shows the variability of database record lengths in each area. By default, it also provides information about the 10 largest database records in each area that are tracked by the analysis process.
You can override the default number of largest database records to be tracked by the analysis process. To request a number of largest database records that is different from the default, specify the desired value on the LARGEST_DATABASE_RECORDS keyword. This keyword can be specified under any primary command that enables you to request analysis with the POINTER_VALIDATION keyword.

The Record Length Analysis Report can be generated only when POINTER_VALIDATION=FULL is specified on the primary command.

The Record Length Analysis Report shows database record lengths in 21 reporting intervals. Each reporting interval provides statistics about the number and percentage of database records and the average, maximum, and minimum record lengths.

**Record Placement Analysis Report**

The Record Placement Analysis Report shows the number of database records in each of seven placement categories and the statistics about the number of DOVF and IOVF blocks used by a record. The report indicates where the data will be placed and overall segment placement distribution. These statistics are helpful in determining sources of high I/O.

The Record Placement Analysis Report can be generated only when POINTER_VALIDATION=FULL is specified.

The Record Placement Analysis Report provides information to help you determine

- when to reorganize a DEDB area
- when to expand control interval (CI) size
- the optimal value for the UOW parameter in the DBD

**Record Profile Analysis Report**

The Record Profile Analysis Report shows statistics about the number and length of database records, dependent segment frequencies, and length statistics. SDEP segments are not included in this report.

The Record Profile Analysis Report provides valuable information about the characteristics of the database records. In addition, the information is used to determine

- when to reorganize an area
- optimal values for the ROOT and UOW parameters in the DBD
Synonym Chain Analysis Report

The Synonym Chain Analysis Report gives detailed information about the frequency and length of synonym chains. The report provides statistics about root segment placement and physical I/O requirements.

The Synonym Chain Analysis Report can be generated only when POINTER_VALIDATION=FULL or RAP_VALIDATION=XREF is specified.

The Synonym Chain Analysis report provides valuable information to determine

- optimal values for ROOT and UOW parameters
- performance tuning for an area
- effectiveness of a randomizing module (In theory, root segments should be distributed evenly across all RAPs.)

Sample offline ANALYZE command scenarios

For scenarios that show how to use the ANALYZE command with key related keywords and subcommands, see Appendix C, “Sample command scenarios.”
Chapter 5 Changing DEDB structure in place

This chapter provides information on the capabilities, setup, and use of the database change function that is provided by the Fast Path Reorg/EP. This function lets you change the structure of a DEDB without unload/reload processing. This chapter discusses the following topics:

- Change function overview .......................................................... 126
- Restructuring options ................................................................. 127
- Preparing for a DEDB change ...................................................... 127
- Change function restrictions ...................................................... 131
- Change function inputs and outputs ......................................... 132
- DBRC considerations ............................................................... 133
- CHANGE command keywords and subcommands ..................... 134
- Specifying input and output ...................................................... 136
  - Specifying areas for input .................................................... 136
  - Specifying areas for output ................................................. 137
- Changing a DEDB by using a generalized randomizer ............... 138
  - Changing the entire area ................................................... 138
  - Changing each area separately .......................................... 139
- Allocating input area data sets .............................................. 140
- Allocating output area data sets .......................................... 140
  - Using DD statements ....................................................... 140
  - Using dynamic allocation to name the output area data set .... 141
  - Using the ALLOCATE subcommand to customize dynamically defined output areas .................................................. 141
- ALLOCATE subcommand sample scenarios ............................... 145
- Controlling free space in IOVF .............................................. 145
- Placing segments with load control ........................................ 146
  - Defining segments for load control placement ....................... 147
  - Placement control using segment content ............................ 148
  - Placement control using segment counts ............................ 150
- Controlling segment compression .......................................... 151
- Selecting SDEP processing method ......................................... 152
- Controlling subset pointer retention ..................................... 154
- Selecting database segments ............................................... 154
  - Including and excluding segments .................................... 154
  - Narrowing the selection .................................................... 156
Change function overview

By using the Fast Path/EP change function, you can change the structure of a DEDB in place without unload/reload processing. Restructuring a DEDB by using the change function yields less I/O, processing time, and downtime than other DEDB restructuring methods.

**NOTE**
The change function restructures DEDBs in offline mode only.

The change function can be used to change the number of areas in a DEDB, resize an area, resize the root addressable area (RAA) of an area, insert or remove segment types, or change the randomizer in a DEDB. If the DEDB uses data compression, you can add, change, or remove the compression.

If you have created an index or indexes to the DEDB by using the BMC Fast Path Indexer/EP product, you can also rebuild any (and all) indexes that are registered for the DEBD at the same time that the change function is performed.
Restructuring options

The Fast Path Offline Suite offers different methods for changing the structure of a DEDB.

**Change**

In most instances, the change function that is provided by Fast Path Reorg/EP is a more efficient method of restructuring a DEDB than unload/reload processing. The change function reads an area directly from DASD or from an image copy and writes the restructured DEDB directly to DASD. Writing directly to DASD potentially can save I/O, processing time and downtime.

**Unload and reload**

Fast Path Reorg/EP also includes unload and reload functions to perform the database change. The unload and reload functions provide significant advantages when compared to other unload/reload processing utilities. For more information, see Chapter 6, “Unloading and Reloading a DEDB.”

**Reorganize and extend**

Another restructuring option offered by Fast Path Reorg/EP is the offline space extend feature that is available with the reorganize function. If an out-of-space condition is imminent in IOVF or SDEP, this feature lets you extend the IOVF and SDEP portions of a DEDB during a reorganization. For more information, see Chapter 3, “Reorganizing a DEDB.”

**Preparing for a DEDB change**

Several tasks are necessary to prepare for a DEDB change:

1. Modify the database definition (DBD) to incorporate the changes.

   Use the DBDGEN and ACBGEN utilities to construct the ACB that contains the modified definition. Be sure to save a copy of the original ACB. The original ACB is required to access the existing area data sets before the DEDB change function is complete.
Examples of DBD redefinition that could occur during database change are as follows:

- modifying area space parameters
- increasing or decreasing the number of areas
- changing a database structure
- adding or deleting segments
- adding or deleting subset pointers

If these changes affect any indexes that are registered to the DEDB that were created using the BMC Fast Path Indexer/EP product, you must incorporate these changes into the index registration.

**NOTE**
If the original DBD contains a definition for an SDEP segment, you must select the technique to be used for processing these segments into the changed database.

2. Make image copies of areas before processing.

   A valid image copy should be made of each original area before executing the change function to ensure that, if problems arise, you can perform adequate recovery of the original area.

3. Decide the type of input to the change.

   You can use the original area data sets or the image copy data sets as input to the change function:

   - When input to the change function is the *original* area data sets, the change function operates in a DASD-to-DASD manner. Sufficient DASD must be available to store the original area data sets and the new area data sets simultaneously. Input area data set names must be different from output area data set names.

   - When input to the change function is the *image copy* data sets, the change function operates in a tape-to-DASD manner. This method reduces the DASD space requirement, but requires that you supply sufficient cache resources to process the input data. For information about using caching with the change function, see “Enhancing performance” on page 163.
4. Make any area data set name alterations.

If you are using an area data set on DASD as input to the change function, the function requires the input area data set name and the output area data set name to be different. If DBRC is active, DBRC registers the change function output data set only. To accommodate naming requirements with DBRC, you can create steps to make the output data set name the same as the name that is registered in DBRC:

- Ensure that an image copy of the original data set has been made.
- Use the VSAM ALTER command to change the original area data set name to a different name.
- Allocate a new data set with the original name. This data set will be used as the output data set of the change function.

When you perform the change function, use the area data set that has the new name assigned as input. For output, use the output area data set that has the same name as the original data set. DBRC registers the output data set. Because the output data set has the same name as the original name, no DBRC name registration changes are required.

5. Make any required DBRC changes.

Consider the implications that your DEDB change will have in regard to DBRC. For example, if you are adding one or more areas to the database, you must add the new areas to the DBRC registrations.

DBRC registers only the output area data set of the change function. For this reason, you might need to consider your input and output data set naming requirements as discussed in step 4.

For information on more DBRC activity that is required before executing the change function, see “DBRC considerations” on page 133.

6. Make any required changes to the randomizer.

Consider the implications that your DEDB change will have in regard to the randomizer. If you are adding one or more areas to the database, for example, you might need to modify the randomizer.

The change function uses the randomizer that is specified by the new DMB in the IMSACB library to select the output area to which the records are written. The randomizer is obtained from the IMSRESLB DD statement, if supplied, or from STEPLIB. The randomizer that is specified by the old DMB in the OLDACB library is not used.
7. Consider the implications of using a generalized randomizer.

If you use a generalized randomizer and any changes are made to the DEDB that affect the number of root anchor points (RAPs), all areas in the DEDB must be input to the change process at the same time. For information about completing the change process when a generalized randomizer is used, see “Changing a DEDB by using a generalized randomizer” on page 138.

8. Consider suppression of repetitious messages.

For each anomaly encountered in the area, the change function generates a message with a specific suffix (severity) level. When the number of places that a condition exists is large, a large number of messages are produced. You can reduce the number of repetitious messages by using the MESSAGE_SUPPRESSION keyword to specify the maximum number of times that any informational, warning, error, or critical message is produced. For the change process, this keyword functions in the same manner as it does for the Fast Path/EP analysis process. For more information, see “Suppressing repetitious messages” on page 111.

9. Consider rebuilding indexes after processing.

If the changes to the DEDB require changes to its registered index (or indexes), you must rebuild the affected index (or indexes). You can specify execution of the index build function as part of the change function or as a separate stand-alone process.

10. Consider analyzing areas after processing.

Analyzing the areas after change processing can ensure that the results of your changes match your expectations. By default, a quick (checksum) analysis is executed as part of the change command if you have a license for the Fast Path Analyzer/EP product. If a more detailed analysis (or no analysis) is desired, you can override the default by specifying the POINTER_VALIDATION keyword with the appropriate parameter value. You can specify execution of the analysis function as part of the change function or as a stand-alone process.

11. Make image copies of areas after processing.

Making an image copy of the areas after change processing and before any applications access the areas is required to ensure that, if problems arise, you can perform adequate recovery. You can specify execution of the image copy function as part of the change function or as a separate stand-alone process.
Change function restrictions

The following restrictions apply to the DEDB change function:

- Only one CHANGE command per job step can be specified, and it must be the only command in the PFPSYSIN input.

- You cannot change segment names of root segments (except in databases consisting only of root segments). You can change other segment names.

- You cannot change the hierarchical path of a dependent segment (cannot insert segments between existing segments).

- You cannot change a direct dependent segment to an SDEP or an SDEP to a direct dependent segment.

- If SMS-managed storage classes are used for initialized areas and the defined cluster spans the area across multiple volumes, you must allocate the area to a guaranteed space storage class. Fast Path Reorg/EP will not format out any portion of the area that resides on a candidate volume.

---

**WARNING**

If you use the CONFIGURE_AREA, CONFIGURE_IOVF, CONFIGURE_RAA or CONFIGURE_SDEP keyword with the ALLOCATE subcommand to dynamically define the output area for a CHANGE process, do not allocate the area to a guaranteed space storage class. If you do so, the guaranteed space storage class will override the configuration specifications.

---

- When a change is made to the key of a dependent segment, the segments must remain in ascending key sequence under the root key. Fast Path Reorg/EP does not have a facility to resequence the dependent segments. A failure of the change process occurs if the change to a dependent segment key is not specified in ascending key sequence.

- If an image copy is used as input to the DEDB change function, a z/OS data space is used. The size of the required data space cannot be greater than 2 GB. If this limit is exceeded, Fast Path Reorg/EP issues an error message and terminates.

---

**NOTE**

The 2 GB limit applies to the combined size of the IOVF and SDEP portion of the database, not the size of the image copy. Fast Path Reorg/EP tries to minimize the size of the required data space by using an internal compression algorithm on the IOVF and SDEP blocks.
When using SDEP_PROCESS=PHYSICAL, you cannot perform any of the following tasks:

— change the CI size of the area
— change the compression parameters for the SDEP segment
— specify the OUTPUT subcommand for the SDEP segment
— modify the randomization parameters such that root segments will be randomized to a different area

**Change function inputs and outputs**

The following sources can be used as input to create a new restructured area:

- DEDB area data sets
- image copies of the DEDB areas

The input source to select for the change function depends on the available resources of your specific processing environment. Using an area data set as input requires enough available DASD to store input and output versions of the area. Using an image copy as input eliminates the need for duplicate DASD but requires the use of cache storage and a tape drive.

As shown in Figure 70, the change function processes input from an area data set or an image copy data set. The change function processes outputs to a new area data set and can create an output image copy of the changed area data set.
During DEDB restructuring, the Fast Path Reorg/EP change function initializes the new DEDB automatically. This initialization saves job steps and further conserves DEDB maintenance resources.

The change function generates a summary report to the report log when DEDB change activities are complete. For an example of the Area Change Report, see the PFPCHNG member in the REPORTS data set.

**DBRC considerations**

When DBRC is active during the execution of the change function, and the area is registered with DBRC, the change function verifies from DBRC that each output area is marked as “Recovery Needed.” The product also obtains authorization for exclusive access (EX) from DBRC for each output area data set. No DBRC authorization is performed on the input data sets. These conditions are shown in Table 9.

**Table 9  DBRC settings for change processing**

<table>
<thead>
<tr>
<th>Command</th>
<th>Authorization</th>
<th>Recovery Needed flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHANGE (inputs)</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>CHANGE (outputs)</td>
<td>exclusive access (EX)</td>
<td>ON</td>
</tr>
</tbody>
</table>
If your areas do not reflect the appropriate “Recovery Needed” condition, then their status must be changed before execution of the change function. To set the “Recovery Needed” flag ON, execute the DBRC utility using the command shown in Figure 71.

**Figure 71  DBRC commands for setting the Recovery Needed flag**

```
CHANGE.DBDS DBD(dbdname) AREA (areaname1) RECOV
```

Multiple area data sets (MADS) are not supported. When executing the change function, the change function creates only the first area data set registered in the ADS list (in collating sequence by DD name). All other area data sets are marked as unavailable. When change processing has completed, use the IBM Online MADS Create utility to resynchronize the other (unavailable) area data sets.

When change processing is completed, the change function resets the “Recovery Needed” flag and notifies DBRC that an image copy is required for each area by setting the “Image Copy Needed” flag. You can use the IC subcommand to request that an image copy be created for each area during change processing. DBRC is informed (NOTIFY.IC) when this image copy is created successfully, which resets the “Image Copy Needed” flag.

If you do not take an image copy during the change process, an informational message is issued to indicate that an image copy must be taken before each area can be updated online. BMC Software recommends that you ensure that a valid image copy of the input area exists before executing the change function. You should also take an image copy of your output area during or after executing the change function.

**NOTE**

The change process also creates a REORG record in DBRC, if DBRC detects that this record is required for recovery.

---

**CHANGE command keywords and subcommands**

Keywords and subcommands that are available for the CHANGE command are listed in Table 10.

<table>
<thead>
<tr>
<th>Function</th>
<th>Command or subcommand</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>selecting the database and areas</td>
<td>CHANGE</td>
<td>DBD, IAREA, OAREA</td>
</tr>
</tbody>
</table>
### Table 10 CHANGE command keywords and subcommands (part 2 of 2)

<table>
<thead>
<tr>
<th>Function</th>
<th>Command or subcommand</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>allocating the input area data sets</td>
<td>CHANGE</td>
<td>INPUT_DSN_MASK</td>
</tr>
<tr>
<td>allocating the output area data sets</td>
<td>CHANGE</td>
<td>OUTPUT_DSN_MASK</td>
</tr>
<tr>
<td>defining the output area data sets</td>
<td>ALLOCATE</td>
<td>all associated keywords</td>
</tr>
<tr>
<td>controlling free space in IOVF</td>
<td>CHANGE</td>
<td>IOVF_LOAD_HWM</td>
</tr>
<tr>
<td>placing segments with load control</td>
<td>LOADCTL</td>
<td>all associated keywords</td>
</tr>
<tr>
<td>controlling segment compression</td>
<td>CHANGE</td>
<td>EXPAND</td>
</tr>
<tr>
<td>selecting SDEP processing method</td>
<td>CHANGE</td>
<td>SDEP_PROCESS</td>
</tr>
<tr>
<td>controlling subset pointer retention</td>
<td>CHANGE</td>
<td>SUBSET_POINTERS</td>
</tr>
<tr>
<td>changing segment contents</td>
<td>OUTPUT</td>
<td>all associated keywords</td>
</tr>
<tr>
<td>selecting database segments for input</td>
<td>INCLUDE EXCLUDE</td>
<td>SAMPLE_INTERVAL SAMPLE_LIMIT SEGMENT WHERE</td>
</tr>
<tr>
<td>controlling processing of root segments to non-processed areas</td>
<td>CHANGE</td>
<td>BYPASS_RECORD</td>
</tr>
<tr>
<td>rebuilding indexes associated with changed DEDB</td>
<td>IX</td>
<td>all associated keywords</td>
</tr>
<tr>
<td>analyzing the DEDB during change</td>
<td>CHANGE</td>
<td>POINTER_VALIDATION RAP_VALIDATION SDEP_VALIDATION LARGEST_DATABASE_RECORDS</td>
</tr>
<tr>
<td>making an image copy during DEDB change</td>
<td>IC</td>
<td>all associated keywords</td>
</tr>
<tr>
<td>enhancing performance</td>
<td>CHANGE</td>
<td>ICACHE OCACHE INDEX_THREADS INPUT_THREADS OUTPUT_THREADS</td>
</tr>
</tbody>
</table>
For more information about the following topics, see the Fast Path/EP Series Reference Manual:

- syntax of commands, subcommands, and keywords that are discussed in this book
- diagrams that show the syntax and available parameters and values for Fast Path Offline Suite commands and subcommands

Specifying input and output

The DBD keyword identifies the name of the DEDB (DBD name) to be changed. The DBD keyword is required for change processing.

Specifying areas for input

The IAREA keyword can be used to select specific areas to be used as input. These areas are described by the OLDACB library. If you omit the IAREA keyword, all areas that are defined in the DEDB are read.

Areas can be specified on the IAREA keyword by using any combination of area names, area numbers, or a area ranges. The following parameters are available for the IAREA keyword:

- IAREA=ALL (default) or IAREA=* specifies all areas of the DEDB.
- IAREA=areaname specifies one or more areas by using the one-character to eight-character area name for each area specified. Multiple area names must be enclosed in parentheses and separated by commas.
- IAREA=areanumber specifies one or more areas by using the one-character to five-character area number for each area specified. Multiple area numbers must be enclosed in parentheses and separated by commas.
- IAREA=(RANGE=(startarea,endarea)) specifies a consecutive range of areas using either areaname or areanumber parameters. The area associated with startarea must be less than the area number associated with endarea.

An asterisk (*) can be used to specify all areas of the DEDB. When the * character is used with the RANGE keyword, it can be used to specify the beginning or ending range for specific areas of the DEDB.

To change an entire DEDB, use a command set like the example shown in Figure 72.
Specifying areas for output

The OAREA keyword can be used to select specific areas to be used as output. These areas are described by the IMSACB library. If you omit the OAREA keyword, all areas that are defined in the DEDB are written to the output.

Areas can be specified on the OAREA keyword by using any combination of area names, area numbers, or a area ranges. The following parameters are available for the OAREA keyword:

- **OAREA=ALL** (default) or **OAREA=*** specifies all areas of the DEDB.
- **OAREA=areaname** specifies one or more areas by using the one-character to eight-character area name for each area specified. Multiple area names must be enclosed in parentheses and separated by commas.
- **OAREA=areanumber** specifies one or more areas by using the one-character to five-character area number for each area specified. Multiple area numbers must be enclosed in parentheses and separated by commas.
- **OAREA=(RANGE=(startarea,endarea))** specifies a consecutive range of areas using either **areaname** or **areanumber** parameters. The area associated with **startarea** must be less than the area number associated with **endarea**.

An asterisk (*) can be used to specify all areas of the DEDB. When the * character is used with the RANGE keyword, it can be used to specify the beginning or ending range for specific areas of the DEDB.
Changing a DEDB by using a generalized randomizer

Values for the OAREA keyword function in the same manner as for the IAREA keyword. See the examples shown for the IAREA keyword in Figure 72 and Figure 73.

Changing a DEDB by using a generalized randomizer

If you use a generalized randomizer with a DEDB, any change in the number of root anchor points (RAPs) affects the distribution of records across the DEDB. Changes that affect the number of RAPs include the following:

- changes to the UOW or ROOT (change in RAA value)
- increasing or decreasing the total number of areas in the DEDB

These changes to the number of RAPs require that all areas in the DEDB be input to the change process at the same time, preferably in the same command set.

The design flexibility of Fast Path Reorg/EP enables parallel processing of DEDBs in multiple ways. The product will not terminate processing if all areas in the DEDB are not defined to the change process when a generalized randomizer is used.

**WARNING**

Failure to include all existing areas as input to the change process at the same time might cause records to be discarded because of re-randomization by the generalized randomizer. In this situation, Fast Path Reorg/EP issues message BMC11178W to indicate a change in the structure of the database which affects the entire DEDB, but that not all areas were included in the process. For example, if the change is processing one area at a time and a record moves from area1 to area2 because of re-randomization and there is no area data set for area2, the entire record is discarded.

Changing the entire area

The simplest method to change a DEDB when a generalized randomizer is used is to execute the change with only the DBD keyword (without the IAREA and OAREA keywords) as shown in Figure 74. By default, all areas are processed by the CHANGE command. Area data sets are dynamically allocated.

**Figure 74** Changing entire area when using generalized randomizer (part 1 of 2)

```plaintext
//PFPMAIN EXEC PGM=PFPMAIN,REGION=0M
//STEPLIB DD DSN=BMC.PFP.LOAD,DISP=SHR
// DD DSN=IMSVS.RESLIB,DISP=SHR
```
Changing each area separately

If you want to execute a separate change process for the data contained in each area, specify the CHANGE command with the OAREA keyword to specify the area name as shown in Figure 75. Repeat this process for each area in the DEDB. Because the input areas can be shared, all jobs can be executed concurrently.

**NOTE**

To select all areas, specify IAREA=ALL, or omit the IAREA keyword which will default to ALL.

---

Figure 75  Changing each area of a DEDB separately when using generalized randomizer

```plaintext
//PFPPMAIN EXEC PGM=PFPPMAIN,REGION=OM
//STEPLIB DD DSN=BMC.PFP.LOAD,DISP=SHR
// // IMSACB DD DSN=IMSVS.NEW.ACBLIB,DISP=SHR
// OLDACB DD DSN=IMSVS.OLD.ACBLIB,DISP=SHR
// IAREA1 DD DSN=PFPP.DB400.AREA1,DISP=OLD
// IAREA001 DD DSN=PFPP.DB400.AREA1.IC,DISP=SHR
// IAREA002 DD DSN=PFPP.DB400.AREA2.IC,DISP=SHR
// IAREA003 DD DSN=PFPP.DB400.AREA3.IC,DISP=SHR
// IAREA004 DD DSN=PFPP.DB400.AREA4.IC,DISP=SHR
// PFPPRINT DD SYSOUT=*  
// PFPSYSIN DD *  
CHANGE DBD=DB400
END
```
Allocating input area data sets

One input data set is required for each area name that is specified using the IAREA keyword. This input data set can be the old area data set or an image copy. The DD statement can be named IAREAxxx, where xxx is the three-digit area number less than 1000, or IARxxxxx, where xxxxx is the five-digit area number from 00001 through 02048. If these DD statements are omitted from the JCL, Fast Path Reorg/EP attempts to dynamically allocate it by using the INPUT_DSN_MASK keyword.

Allocating output area data sets

You can allocate the output area for the CHANGE command by using traditional DD statements or by using dynamic allocation. When using dynamic allocation, you can also specify certain processing criteria and allocation characteristics for the output area.

Using DD statements

You can use the following DD statements to allocate the output area for the CHANGE command.

Areaname

The areaname DD statement identifies the area data set to be created. If the areaname DD statement is omitted from the JCL, Fast Path Reorg/EP attempts to dynamically allocate it.

The area name that is specified on the areaname DD statement must be the same as the area name that is specified in the ACB library. Fast Path Reorg/EP verifies this match by using the IMSACB DD statement.

PFPTOTAL

The PFPTOTAL DD statement allocates an optional output PFPTOTAL data set. This output data set can be used as an audit file to verify segment output totals after using the change process.

The PFPTOTAL data set is designed for input to a user-written program. For more information about the PFPTOTAL DD statement and the record format, see the Fast Path/EP Series Reference Manual.
Using dynamic allocation to name the output area data set

If you are using dynamic allocation to define the output area, do not include the areaname DD statement. Fast Path Reorg/EP attempts to obtain the data set name for allocation in the following order:

1. If the OUTPUT_DSN_MASK keyword is specified, it is used to generate the data set name.

2. If DBRC is active and the area is registered, the registered area data set name is obtained from DBRC.

3. The STEPLIB is searched for the DFSMDA member that contains the data set name for this area.

If DBRC is active and the area is registered with DBRC, the allocated data set name must match the registered data set name, regardless of how it is allocated.

Using the ALLOCATE subcommand to customize dynamically defined output areas

If you elect to dynamically define the output areas for the CHANGE command, you can specify unique allocation characteristics for the output areas by using the ALLOCATE subcommand and its related keywords. The ALLOCATE subcommand can be used as a replacement for DELETE, DEFINE, and other statements that are normally specified by executing the Access Method Services IDCAMS program.

NOTE

If the IDCAMS program is already defined in your job input, it is not necessary to remove it. However, any defining statements that are specified under IDCAMS for the output area are superseded by similar parameters that are specified on the ALLOCATE subcommand.

By applying the functionality available with the ALLOCATE subcommand, you can accomplish the following tasks and performance improvements:

- automatically adjust the VSAM cluster definition to match an area’s size characteristics
- help to simplify JCL streams and job restarts by combining VSAM cluster definition (traditionally defined by IDCAMS) and area processing into a single step
- improve the area’s usage of fragmented DASD space and irregular sized extents
Using the ALLOCATE subcommand to customize dynamically defined output areas

- enhance system I/O performance by promoting parallel processing against the area by easily separating the RAA, IOVF, and SDEP portions of an area across multiple volumes

The ALLOCATE subcommand lets you perform the following tasks:

- delete/define, reuse, or rename the VSAM cluster that is used for the output areas
- pass a sequential or partitioned data set to Access Method Services to be used as input for allocation of the new VSAM cluster
- specify the number of volumes and allocation units for the output areas
- specify the number of volumes and allocation units for the RAA, IOVF, and SDEP portions of the output areas
- pass optional parameters to Access Method Services for the VSAM cluster definition
- specify the SMS storage class, management class, and data class for the output areas
- specify space requirements for the output areas
- specify volume serial identifiers for the output areas

These functions are defined by specifying combinations of keywords that are specified on the ALLOCATE subcommand.

**Identifying the areas for ALLOCATE processing**

The OAREA keyword is specified on the ALLOCATE subcommand to identify the areas for which you want to customize allocation. You can specify one or more areas on a single ALLOCATE subcommand, or specify separate ALLOCATE subcommands for any (or all) output areas that are identified on the CHANGE command’s OAREA keyword.

Areas can be specified on the OAREA keyword under the ALLOCATE subcommand by using any combination of area names, area numbers, or a area ranges. For a description of valid OAREA keyword parameters, see “Specifying areas for output” on page 137.
Specifying the area to be used

The ACTUATE keyword requests optional VSAM cluster processing. The following parameters are available:

- **ACTUATE=DELETE** causes the current VSAM cluster to be deleted before the new VSAM cluster is allocated.
- **ACTUATE=REUSE** causes the current VSAM cluster to be allocated and overwritten without allocating a new VSAM cluster.
- **ACTUATE=RENAME** causes the current VSAM cluster name to be altered according to the data set name mask that you specify before allocating the new VSAM cluster.
- **ACTUATE=IDCAMS** causes a sequential or partitioned data set to be passed to Access Method Services as input for the allocation of the new VSAM cluster. Due to the design of this feature, each area should have a separate IDCAMS definition.

If you omit the ACTUATE keyword, it is assumed that the output area does not already exist. A new output area is allocated.

You can also specify the EROPT parameter on the ACTUATE keyword to specify whether ALLOCATE subcommand processing should continue if a processing error is encountered. Specify EROPT=IGNORE to continue processing. Specify EROPT=ABORT to terminate processing.

Allocating the output area and specific portions of the area

Four keywords are available on the ALLOCATE subcommand that control the following allocation characteristics for the specified output area or portion of the output area:

- size of each extent for the area or portion of the area
- number of extents (volumes) that the area or portion of the area will span

By default, the entire output area or any portions of an output area (RAA, IOVF, or SDEP) that are not explicitly specified with extent size or volume count parameters will be contained on one volume.

The size of the extents are controlled by specifying one or more values on the following keywords:

- The CONFIGURE_AREA keyword specifies the size of the extent or extents for the entire area data set.
The CONFIGURE_RAA keyword specifies the size of the extent or extents for the root addressable portion of the area data set.

The CONFIGURE_IOVF keyword specifies the size of the extent or extents for the independent overflow portion of the area data set.

The CONFIGURE_SDEP keyword specifies the size of the extent or extents for the sequential dependent portion of the area data set.

The values that are specified on the CONFIGURE_AREA keyword or any combination of the CONFIGURE_RAA, CONFIGURE_IOVF and CONFIGURE_SDEP keywords represent the amount of track or cylinder space to be allocated to the extent. Selection of tracks or cylinders for the extents is requested by specifying the SPACE keyword parameter on the ALLOCATE subcommand or is determined by IDCAMS.

To define multiple extents with the same amount of space, you can specify a value on the VOLCNT parameter with an extent size value. The value that is specified on the VOLCNT parameter indicates the number of extents (volumes) that the area or portion of the area will span. Each extent is sized according to the specified extent size.

You can specify the IDCAMS_OPTION keyword on the ALLOCATE subcommand to pass optional parameters to Access Method Services for the VSAM cluster definition. Numerous parameters are available for use with this keyword to control the values that are used by AMS for the VSAM cluster definition. For more information, see the IBM Access Method Services Reference Manual.

Specifying space requirements, class, and volume identifier for the output area

Other keywords let you specify the SMS class, space requirements, and volume serial identifier for the output area:

- AVGREC
- DATAACLAS
- MGMTCLAS
- SPACE
- STORCLAS
- VOLSER

These keywords function in the same manner as on other Fast Path/EP subcommands.
ALLOCATE subcommand sample scenarios

For scenarios that show how this subcommand can be used with key related keywords to define allocation parameters for areas that are output by CHANGE, RELOAD, and INITIALIZE commands, see Appendix C, “Sample command scenarios.”

Controlling free space in IOVF

When IOVF space is needed to accommodate a CHANGE process, the normal space search algorithm places segments sequentially into IOVF blocks. The placement of segments within IOVF in this manner can affect application performance by causing longer reads of IOVF to find available free space. It is sometimes useful to influence the space search algorithm during the change function to control the amount of space that is used in each IOVF block.

You can request a percentage of total space to be used in each IOVF block when the CHANGE command reloads segments for each root. The IOVF_LOAD_HWM keyword can be specified to request a percentage value between 50 and 100 (the default) of space to be filled in each IOVF block. During CHANGE command processing, Fast Path Reorg/EP fills each IOVF block to the specified percentage before attempting to place segments in new storage locations.

If you request a value (percentage) less than 100 and the IOVF is not large enough to store all data at that fill level, Fast Path Reorg/EP makes a second load pass to load data until all data is stored or all IOVF blocks in the area are full.

The example in Figure 76 requests that each block of IOVF be filled to 70 percent of its space capacity during reloading of the area. If 70 percent of each IOVF block is not adequate for the amount of IOVF needed by the CHANGE, Fast Path Reorg/EP makes a second pass through IOVF and fills each sequential IOVF to 100 percent of capacity until CHANGE command processing is completed. If all existing space in IOVF is used on the second pass, message BMC111167C is issued to indicate the out-of-space condition.

Figure 76  Sample CHANGE using IOVF_LOAD_HWM keyword

```
CHANGE DBD=dbname, IAREA=areaname
   IOVF_LOAD_HWM=70
```
Placing segments with load control

The normal space search algorithm places segments into the control intervals in a tightly packed manner. The root segment is placed within the RAA block (if room is available) that is determined by the randomizer. Direct dependent segments belonging to the root are placed into the same RAA block or DOVF blocks within the UOW until they are all used. IOVF blocks are selected and filled as needed.

The placement of segments within the types of control intervals (RAA, DOVF, and IOVF) has a significant effect on performance. It is sometimes useful to influence the space search algorithm during the DEDB change function to relegate infrequently used segments into IOVF (or DOVF) blocks. Relegating infrequently used segments preserves space in the RAA (or DOVF) blocks for more active data.

Placement of selected segments during the change function can be accomplished by using the LOADCTL subcommand and its related keywords. The LOADCTL subcommand lets you perform the following tasks:

- select a segment, a segment and its dependents, or its dependents only for placement
- control the placement of selected segments into IOVF or DOVF
- control placement of segments, based on their data content
- control placement of segments, based on a specified number of segment occurrences

**NOTE**
To maintain this data placement, you must continue to use the same LOADCTL subcommand (or subcommands) for any subsequent change processing on the DEDB.

Figure 77 shows a typical DEDB hierarchy. This hierarchy is used in subsequent examples in this section to show how the LOADCTL subcommand can be used with its associated keywords.
Defining segments for load control placement

The SEGMENT keyword is required with the LOADCTL subcommand to identify segments for load control placement. The following subparameters are available for the SEGMENT keyword to provide selection versatility:

- **SEGMENT=(segname,ONLY)** (the default) specifies that the LOADCTL subcommand applies to the named segment only. Its dependent segments (if any) are not affected.

- **SEGMENT=(segname,DEPENDENTS)** specifies that the LOADCTL subcommand applies to the dependents of the named segment only. The named segment is not affected.

- **SEGMENT=(segname,BOTH)** specifies that the LOADCTL subcommand applies to the named segment and to all of its dependents.

The LOCATION keyword specifies where you want the segments placed:

- **LOCATION=IOVF** (the default) places segments in IOVF storage, regardless of the amount of space that is available in RAA and/or DOVF blocks.

- **LOCATION=DOVF** places segments in DOVF storage, regardless of the amount of space that is available in the RAA block. If there is not enough space in the DOVF blocks, the segment is placed into IOVF.

**Placing a specific segment type**

If you specify the SEGMENT keyword with the ONLY subparameter (the default), the named segment only is placed by using load control.

In the example in Figure 78, all occurrences of SEGB are placed into IOVF.

---

**Figure 77  Database hierarchical structure in LOADCTL examples**

![Database hierarchical structure diagram]

---

**Defining segments for load control placement**

The SEGMENT keyword is required with the LOADCTL subcommand to identify segments for load control placement. The following subparameters are available for the SEGMENT keyword to provide selection versatility:

- **SEGMENT=(segname,ONLY)** (the default) specifies that the LOADCTL subcommand applies to the named segment only. Its dependent segments (if any) are not affected.

- **SEGMENT=(segname,DEPENDENTS)** specifies that the LOADCTL subcommand applies to the dependents of the named segment only. The named segment is not affected.

- **SEGMENT=(segname,BOTH)** specifies that the LOADCTL subcommand applies to the named segment and to all of its dependents.

The LOCATION keyword specifies where you want the segments placed:

- **LOCATION=IOVF** (the default) places segments in IOVF storage, regardless of the amount of space that is available in RAA and/or DOVF blocks.

- **LOCATION=DOVF** places segments in DOVF storage, regardless of the amount of space that is available in the RAA block. If there is not enough space in the DOVF blocks, the segment is placed into IOVF.

**Placing a specific segment type**

If you specify the SEGMENT keyword with the ONLY subparameter (the default), the named segment only is placed by using load control.

In the example in Figure 78, all occurrences of SEGB are placed into IOVF.
Placing dependents of a segment type

If you specify the SEGMENT keyword with the DEPENDENTS subparameter, all dependents of the named segment are placed by using load control. The named segment is placed normally.

In the example in Figure 79, the DEPENDENTS subparameter is specified. Segment placement control is applied to all dependent segment types of the named segment (SEGB). All dependent segment occurrences (SEGC, SEGD, SEGE, and SEGF) are placed within an IOVF block, regardless of the amount of space that might remain within the RAA and DOVF blocks.

Placing a segment and its dependents

If you specify the SEGMENT keyword with the BOTH subparameter, the named segment and all of its dependents are placed by using load control. In the example in Figure 80, the BOTH subparameter is specified. Segment placement control is applied to SEG and its dependent segment (SEGH).

Placement control using segment content

You can specify that placement control is to be applied, based on the data values that are contained in segments, by specifying the WHERE keyword. The WHERE keyword lets you specify field names or field positions and lengths, an operand, a value, and optional Boolean operators. For information about the operations and syntax of the WHERE keyword and detailed syntax rules for coding these expressions, see the Fast Path/EP Series Reference Manual.
Placing specific segments based on segment data

You can specify that placement control is to be applied, depending on a value or values that appear within the named segment data. The example in Figure 81 requests that placement control be applied to the segment type specified explicitly by the SEGMENT keyword. Selection criteria is specified by using the WHERE keyword.

Figure 81  Sample LOADCTL basic selection criteria

<table>
<thead>
<tr>
<th>CHANGE DBD=dbdname</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOADCTL SEGMENT=(SEGB,ONLY),LOCATION=IOVF,</td>
</tr>
<tr>
<td>WHERE=(3:1 EQ 'X')</td>
</tr>
</tbody>
</table>

If the criteria is met, SEGB is placed within an IOVF block, regardless of the amount of space that might remain within the RAA and DOVF blocks. If the criteria is not met, the segment occurrence is placed normally.

Placing specific segments based on segment data in a different segment

You can specify that criteria be based on data within a different segment with the WHERE keyword by using a qualified field reference. The qualified field must lie within the segment parentage path from the root segment to the segment that is specified by using the SEGMENT keyword.

The example in Figure 82 requests that placement control be applied to SEGD specified by the SEGMENT keyword. Selection criteria is specified by using the WHERE keyword that references a field within the parent segment (SEGB). If the criteria is met, the SEGD segment occurrences are placed within an IOVF block, regardless of the amount of space that might remain within the RAA and DOVF blocks. If the criteria is not met, the SEGD segments are placed normally.

Figure 82  Sample LOADCTL using qualified selection criteria

<table>
<thead>
<tr>
<th>CHANGE DBD=dbdname,IAREA=areaname</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOADCTL SEGMENT=SEGD,LOCATION=IOVF,</td>
</tr>
<tr>
<td>WHERE=(SEGB.3:1 EQ 'X')</td>
</tr>
</tbody>
</table>

Placing multiple segments based on different criteria

You can specify as many LOADCTL subcommands as are required to obtain the desired segment placement.

Figure 83 requests that placement control be used for four segments. The WHERE keyword is used for each segment to specify a Boolean expression that controls the placement of each segment occurrence.
Placement control using segment counts

If multiple LOADCTL subcommands are specified for the same segment type, they are evaluated in the order in which they are specified. The first LOADCTL subcommand for the segment for which the WHERE criteria is met (or if no WHERE criteria is specified) is used to control placement.

Figure 84 shows an example of multiple LOADCTL subcommands that are specified for the same segment. If the first WHERE criteria is met, SEGB is placed in DOVF. If the second WHERE criteria is met, SEGB is placed in IOVF. If neither WHERE criteria is met, the segment is placed normally.

Figure 84  Sample LOADCTL using multiple WHERE statements for the same segment type

CHANGE DBD=dbname, IAREA=areaname
LOADCTL SEGMENT=SEGB, WHERE=(3:2 EQ '80'), LOCATION=IOVF,
LOADCTL SEGMENT=SEGB, WHERE=(3:2 GE '80'), LOCATION=DOVF

Placement control using segment counts

You can specify that placement control be applied after a number of segment occurrences for a root segment have been processed by using the INSERT_LIMIT_COUNT keyword.

The example in Figure 85 requests that placement control be used for a specified segment type. The first five segments of this type for each root segment are placed normally. The sixth and all subsequent occurrences of the segment type are placed within an IOVF block, regardless of the amount of space that might remain within the RAA and DOVF blocks within the UOW.
The example in Figure 86 combines the WHERE and INSERT_LIMIT_COUNT keywords on a single LOADCTL subcommand. All segment types that do not meet the WHERE criteria are placed normally. Of the segments that do meet the WHERE criteria, the first three are placed normally. The fourth and all subsequent occurrences of the segment type that meet the WHERE criteria are placed within an IOVF block, regardless of the amount of space that might remain within the RAA and DOVF blocks within the UOW.

Figure 87 shows an example of the EXPAND keyword with the CHANGE command.

**Figure 87 Using the EXPAND keyword with CHANGE**

```plaintext
CHANGE DBD=dbdname,EXPAND=YES
```
Figure 88 shows an example of using the EXPAND keyword to selectively specify segment expansion and recompression. Only segment1 and segment3 are expanded and recompressed; all other segments are written to the output area unchanged.

**Figure 88  Selectively specifying segments for expansion with CHANGE**

```
CHANGE DBD=dbname,IAREA=ALL,
    OAREA=ALL,EXPAND=(segment1,segment3)
```

### Selecting SDEP processing method

If the input database has an SDEP segment defined, you must indicate how these segments are to be processed. Multiple methods are available for processing SDEP segments during execution of the change function. The method is determined by the SDEP_PROCESS keyword:

- **SDEP_PROCESS=LOGICAL** causes SDEP segments to be processed with the root segment that owns them. The SDEP segments in each output area are physically reordered into the same sequence as the parent root segments. The SDEP segments retain the same logical order (entry sequence) within each database record. If you are running IMS version 6.1 or higher, all SDEP segments are timestamped. If you do not want the SDEP segments to be timestamped, specify the SDEP_PROCESS=V5COMP value described on page 153.

**WARNING**

If your application relies on the marker segment concept, do not use SDEP_PROCESS=LOGICAL.

- **SDEP_PROCESS=PHYSICAL** causes SDEP segments to be processed in the physical sequence in which they appear in the input areas. SDEP segments are inserted into the output areas in the same relative physical location, which fully supports the marker segment concept. They also retain the same logical order (entry sequence) within each database record, including the same cycle count and timestamp.
Selecting SDEP processing method

**NOTE**

The following restrictions and considerations apply when SDEP_PROCESS=PHYSICAL is specified on the CHANGE command:

- You cannot *reduce* the CI size of the area; however, you can *increase* the CI size.
- You cannot modify the compression parameters for the SDEP segment.
- You cannot specify the OUTPUT subcommand for the SDEP segment.
- You cannot modify the randomization parameters such that root segments will be randomized to a different area.
- SDEP pointer errors will not be detected. Consequently, any SDEP errors will not be applied to the error detection count value that you specify on the ERROR_THRESHOLD keyword.

SDEP_PROCESS=V5COMP causes SDEP segments to be processed in conjunction with the root segment that owns them. The SDEP segments in each output area are physically reordered into the same sequence as the parent root segments. All SDEP segments are not timestamped, regardless of the version of IMS you are running. The SDEP segments retain the same logical order (entry sequence) within each database record.

**WARNING**

If your application relies on the marker segment concept, *do not* use SDEP_PROCESS=V5COMP.

SDEP_PROCESS=NONE causes SDEP segments in the input areas to be ignored; these segments are not placed in the output areas. If SDEP segment types are detected in the input area, a warning message is generated to indicate that SDEP segments will be removed.

Figure 89 shows an example of the SDEP_PROCESS keyword with the CHANGE command.

**Figure 89 Using the SDEP_PROCESS keyword with CHANGE**

```plaintext
CHANGE DBD=dbname, SDEP_PROCESS=PHYSICAL
```
Controlling subset pointer retention

You can retain the settings for subset pointers during the change function. This option is determined by the SUBSET_POINTERS keyword. If subset pointers are defined in the input DEDB, you must specify a value for the SUBSET_POINTERS keyword:

- SUBSET_POINTERS=NO causes all subset pointers in the output areas to be cleared. Subset pointers from the input areas are not transferred.
- SUBSET_POINTERS=YES causes the values of subset pointers from the input areas to be transferred to the output areas. The actual pointer value (RBA) is translated so that it refers to the new location of the segment after it has been placed into the output area.

Figure 90 shows an example of the SUBSET_POINTERS keyword with the CHANGE command.

Figure 90 Using the SUBSET_POINTERS keyword

```plaintext
CHANGE DBD=dbdname,SUBSET_POINTERS=YES
```

Selecting database segments

Segments that are selected during input processing are forwarded to output processing for segment formatting. By default, all root segments and their dependent segments are selected, excluding SDEP segments. You can modify the selection of segments by using the optional INCLUDE and EXCLUDE subcommands. You can also use several keywords with the INCLUDE and EXCLUDE subcommands to further refine the selection process.

NOTE

To control SDEP segment selection, use the SDEP_PROCESS keyword discussed in “Selecting SDEP processing method” on page 152.

Including and excluding segments

The INCLUDE subcommand lets you specify criteria for the selection of segments that will be included in the change process. If the INCLUDE subcommand selection criteria are met, the selected segments and their dependents are written to the new area.
The EXCLUDE subcommand lets you specify criteria for selection of segments that will be omitted from the change process. If the EXCLUDE subcommand selection criteria are met, the selected segments and their dependents are not written to the new area.

**NOTE**

You cannot include a dependent segment and exclude its parent segment. Specifying a parent segment on the EXCLUDE subcommand also excludes all dependent segments.

The SEGMENT keyword specifies the name of the segment to which the selection criteria are applied. Additional selection criteria can be specified by using the WHERE keyword, which uses field names or positional identifiers with relational or Boolean logical expressions.

A simple INCLUDE or EXCLUDE subcommand uses the required SEGMENT keyword *without* additional keywords. Understanding the following rules that apply to basic INCLUDE and EXCLUDE subcommands is prerequisite to using additional keywords with INCLUDE and EXCLUDE:

- By default, if no INCLUDE or EXCLUDE statement is used with the CHANGE command, all roots and dependents are selected for processing.

- The INCLUDE subcommand selects the segment that is identified by the SEGMENT keyword and all dependents of the segment. Including a segment in this manner also selects the parent hierarchy chain above the segment.

- The EXCLUDE subcommand excludes the segment that is identified by the SEGMENT keyword and all dependents of the segment. You cannot include a segment and exclude a root or a parent of the segment. Excluding a parent excludes the entire hierarchy beneath the parent.

A basic CHANGE command with an INCLUDE and EXCLUDE subcommand is shown in Figure 91.

**Figure 91 Using the CHANGE command with INCLUDE and EXCLUDE subcommands**

```
CHANGE DBD=dbdname, IAREA=(areanam1,areanam3),
   OAREA=(areanam1,areanam3)
INCLUDE SEGMENT=segname1
EXCLUDE SEGMENT=segname2
```
Narrowing the selection

While identifying the segments of interest for the change process using the INCLUDE and EXCLUDE subcommands, you can apply additional criteria to narrow the selection by using the following keywords:

- The WHERE keyword selects segments by applying conditional criteria that is specified in a Boolean expression.

- The SAMPLE_INTERVAL keyword selects segments by intervals of occurrence.

- The SAMPLE_LIMIT keyword sets a limit on the number of segments meeting the criteria that will be output to the new area.

Conditional criteria

You can narrow segment selection by using the WHERE keyword. This combination applies relational and Boolean operators to apply conditional criteria when selecting segments. For information about the operations and syntax of the WHERE keyword and detailed syntax rules for coding these expressions, see the Fast Path/EP Series Reference Manual.

Figure 92 shows a typical DEDB segment hierarchy. This hierarchy is used in subsequent examples in this section that show how the INCLUDE and EXCLUDE subcommands can be used with their associated keywords.

Figure 93 shows a typical DEDB segment hierarchy. This hierarchy is used in subsequent examples in this section that show how the INCLUDE and EXCLUDE subcommands can be used with their associated keywords.

Figure 92  Typical DEDB segment hierarchy

The example in Figure 93 is used to create a test system database from a production version by selecting a specific set of records. The example selects only the SEGA segments that have the field ROOTKEY equal to BMCPFP. For the selected SEGA segments, only SEGB (and its dependents) where position 23 is equal to 22 will be selected. Only SEGE segments (of the selected SEGA segments) that were created after the year 1997 are selected. Although SEGG is defined in the DBD, it is excluded from the change process.
Narrowing the selection

Chapter 5 Changing DEDB structure in place 157

Criteria that are applied to alternate segments

You can specify segment selection, based on criteria that are applied to alternate segments, by using a segment-qualified field reference within the WHERE keyword. Only the parentage segments within the hierarchical path from the root might be used for qualification.

In the example in Figure 94, SEGC is selected only if FIELD1 in SEGB is equal to the character string BMC.

Segment sampling

The SAMPLE_INTERVAL keyword can be used to specify segment selection based on the frequency of the segment occurrence. For example, if you have specified SAMPLE_INTERVAL=5, every fifth segment will be selected. For a root segment, every fifth occurrence in the database and all of its dependents will be selected. For a dependent segment, every fifth occurrence under its parent is selected.

The SAMPLE_LIMIT keyword can be used to specify the maximum number of segment occurrences to be selected. For example, if you have specified SAMPLE_LIMIT=500, the first 500 segments will be selected. For a root segment, the first 500 occurrences and all its dependents are selected. For a dependent segment, the first 500 occurrences within its parent are selected.

In the example in Figure 95, all areas are included in the process since the IAREA and OAREA keywords are not specified. The result of the INCLUDE subcommand is that every ninth occurrence of SEGA (and all its dependents) are written to the new area with a maximum of 5000 occurrences of SEGA. The EXCLUDE subcommand changes the result of the previous INCLUDE for dependent segment SEGE. For each selected

Figure 93 CHANGE command using INCLUDE, EXCLUDE, and WHERE

```sql
CHANGE DBD=dbdname, IAREA=(areanam1,areanam3),
  OAREA=(areanam1)
INCLUDE SEGMENT=SEGA,
  WHERE=(ROOTKEY EQ 'BMCPFP')
INCLUDE SEGMENT=SEGB,
  WHERE=(23:2 EQ '22')
INCLUDE SEGMENT=SEGE,
  WHERE=(DATE GT '1997')
EXCLUDE SEGMENT=SEGG
```

Figure 94 Qualifying field references using WHERE keyword

```sql
CHANGE DBD=dbdname, IAREA=(areanam1,areanam3),
  OAREA=(areanam1,areanam3)
INCLUDE SEGMENT=SEGC,
  WHERE=(SEGB.FIELD1 EQ 'BMC')
```

Figure 95 Segment sampling

```sql
CHANGE DBD=dbdname, IAREA=(areanam1,areanam3),
  OAREA=(areanam1,areanam3)
INCLUDE SEGMENT=SEGA,
  WHERE=(ROOTKEY EQ 'BMCPFP')
INCLUDE SEGMENT=SEGB,
  WHERE=(23:2 EQ '22')
INCLUDE SEGMENT=SEGE,
  WHERE=(DATE GT '1997')
EXCLUDE SEGMENT=SEGG
```
SEGA segment, the first five segment occurrences for segment SEGE are omitted. If, for example, SEGE has eight occurrences, the last three segment occurrences would be written to the new area. If segment SEGE had only four occurrences under another selected SEGA, no SEGE segments (or their dependents) would be written to the new area.

Figure 95  CHANGE command with SAMPLE_INTERVAL and SAMPLE_LIMIT keywords

```
| CHANGE  DBD=dbname,     |
| INCLUDE SEGMENT=SEGA,   |
|      SAMPLE_INTERVAL=9, |
|   SAMPLE_LIMIT=5000     |
| EXCLUDE SEGMENT=SEGE,   |
|     SAMPLE_LIMIT=5      |
```

Changing segment contents

The OUTPUT subcommand and its associated keywords can be used to change the content of the segment. By default, the output areas will contain the full, unmodified contents of the input segments.

To modify the segment contents, you can use the OUTPUT subcommand with the FIELDS keyword. The FIELDS keyword specifies a list of expressions; the value of each expression is evaluated and placed into the output segment in the order that the expressions are specified. The length of the segment is computed by the product as the sum of the lengths of the values produced from these expressions. For information about the operations and syntax of the FIELDS keyword and syntax rules for coding expressions, see the Fast Path/EP Series Reference Manual.

**WARNING**

When the FIELDS keyword is used with the OUTPUT subcommand to change the key of a dependent segment, the segments must remain in ascending key sequence under their parent. Fast Path Reorg/EP does not have a facility to resequence the dependent segments. The change process terminates if the dependent segments are not maintained in ascending key sequence.

**NOTE**

When the FIELDS keyword is used with the OUTPUT subcommand to modify the content of a compressed segment, you must expand the segment by using the EXPAND=YES value or the EXPAND=(segment1,segment2,...,segmentn) syntax on the CHANGE command. This expansion will ensure that the operands which are specified on the FIELDS keyword correspond to the proper segment columns.

Figure 96 shows an example where a literal value is being inserted into the output segment during a DEDB change.
Samples of INCLUDE, EXCLUDE, and OUTPUT subcommands

Table 11 shows examples of the CHANGE command combined with the INCLUDE, EXCLUDE, and OUTPUT subcommands and the resulting selected segments. These examples use the segment hierarchy that is shown in Figure 92 on page 156.

<table>
<thead>
<tr>
<th>Command Set</th>
<th>Selected Segments</th>
<th>Output Segments</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHANGE IAREA=ALL</td>
<td>All roots and their dependent segments are selected.</td>
<td>A segment record is written for each selected segment.</td>
</tr>
<tr>
<td>no INCLUDE, EXCLUDE, or OUTPUT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>subcommands specified</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXCLUDE SEGMENT=SEGA</td>
<td>No segments are selected.</td>
<td>No segment records are output.</td>
</tr>
<tr>
<td>INCLUDE SEGMENT=SEGB</td>
<td>No segments are selected. (Excluding a parent segment</td>
<td>No segment records are output.</td>
</tr>
<tr>
<td></td>
<td>excludes all of its dependent segments.)</td>
<td></td>
</tr>
</tbody>
</table>

Figure 96 Modifying output segment contents

CHANGE DBD=dbname,
OUTPUT SEGMENT=SEGD,
FIELDS=(3:20,4P’0’,23:*)

The content of the output segment can be controlled by using conditional qualification that this specified by the optional WHERE keyword on the OUTPUT subcommand. If multiple OUTPUT subcommands are specified for the same segment, the first subcommand in which the selection criteria is satisfied is used. If the selection criteria is not satisfied for any OUTPUT subcommand for a segment, the input segment is copied to the output area without modification.

Figure 97 shows an example where the content of a segment is conditionally modified during a DEDB change. If the WHERE condition specified on the OUTPUT subcommand is met, the FIELDS keyword modifies the segment content. If the condition is not met, the segment content is unchanged.

Figure 97 Using WHERE keyword to conditionally specify segment content

CHANGE DBD=dbname,
OUTPUT SEGMENT=SEGD,
FIELDS=(3:20,C’ABC’,26:*)
WHERE=(23:3 EQ C’ ‘)
Processing considerations for randomization of root segments to unprocessed areas

When the CHANGE command is executed under normal conditions, processing continues if any root segments are randomized to an area in the DEDB that is not being processed by the command set and therefore bypassed and uncopied.

The optional BYPASS_RECORD keyword can be used to terminate the change function when root segments are randomized to an area in the DEDB that is not being processed by the command set.

The following values are available for the BYPASS_RECORD keyword:

- **BYPASS_Record=NO** causes processing to terminate if any root segment is randomized to an area in the DEDB that is not being processed. An error message is generated that contains the key of the bypassed root segment.

- **BYPASS_Record=YES** (default) enables the change function to continue processing when a root segment is randomized to an area in the DEDB that is not being processed. An informational message is generated by the change function that includes a count of the total number of bypassed records.

In the example shown in Figure 98, any root segments that are randomized to an area other than *areanam1*, *areanam4*, or *areanam5* due to changes in the DBD or randomizer will cause the change function to terminate.

---

**Table 11  Using INCLUDE, EXCLUDE and OUTPUT subcommands with the CHANGE command (part 2 of 2)**

<table>
<thead>
<tr>
<th>Command Set</th>
<th>Selected Segments</th>
<th>Output Segments</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCLUDE SEGMENT=SEGB</td>
<td>SEGA, SEGE, SEG, and SEGG are selected.</td>
<td>A segment record is written for each selected segment.</td>
</tr>
<tr>
<td>INCLUDE SEGMENT=SEGA, WHERE=(selection criteria)</td>
<td>All SEGA segments that meet criteria and their dependents are selected.</td>
<td>A segment record is written for each selected segment.</td>
</tr>
<tr>
<td>EXCLUDE SEGMENT=SEGB</td>
<td>SEGA and SEGG are selected.</td>
<td>All SEGA and SEGG segments selected are written.</td>
</tr>
<tr>
<td>OUTPUT SEGMENT=SEGG</td>
<td>All roots and their dependent segments are selected.</td>
<td>All segments are written.</td>
</tr>
<tr>
<td>INCLUDE SEGMENT=SEGB, WHERE=(selection criteria) EXCLUDE SEGMENT=SEGC</td>
<td>All SEGA, SEGE, SEG, and SEGG segments are selected. Only SEGB (and SEGD children) segments that meet the INCLUDE selection criteria are selected.</td>
<td>All SEGA, SEGE, SEG, and SEGG segments are written. All SEGB (and SEGD children) segments that meet the selection criteria are written.</td>
</tr>
</tbody>
</table>
If the DEDB that is being changed has registered indexes that were created using the BMC Fast Path Indexer/EP product, you can request that any (or all) of these indexes be rebuilt simultaneously with the change function.

Specify the IX subcommand and associated keywords for the index to be rebuilt. For details about the IX subcommand, see the Fast Path Indexer/EP User Guide.

The example in Figure 99 requests that all indexes which are registered to the DEDB be rebuilt during the change process.

The INDEX_THREADS keyword can be specified on the CHANGE command to specify the number of index threads to be used to rebuild indexes that are identified by an IX subcommand. Figure 100 requests that five threads be used for rebuilding all indexes registered to the source database during the change process.

For more information about the INDEX_THREADS keyword, see the Fast Path Indexer/EP User Guide.

Analyzing the DEDB during change

If your site has a license for the Fast Path Analyzer/EP product, pointer validation will occur automatically (by default) when you execute a database change. This automatic process applies the default value of QUICK for the POINTER_VALIDATION keyword to perform a checksum validation of pointers for each segment type within each UOW of the database that is specified on the CHANGE command.
Automatic pointer validation provides an assurance of the area’s pointer integrity, while providing statistics that show how the database change process affected space usage.

**NOTE**
You can override automatic quick pointer validation and request more detailed validation by specifying \texttt{POINTER VALIDATION=FULL} on the \texttt{CHANGE} or \texttt{GLOBAL} command. Direct pointers are validated by using the cross-reference technique. You can also specify any other valid value for the \texttt{POINTER VALIDATION} keyword (such as \texttt{POINTER VALIDATION=NONE} to completely disable the analysis function).

If an SDEP segment is defined for the database, SDEP pointers are also validated by default using the same technique as specified for direct pointers. You can manually specify another valid value for the \texttt{SDEP VALIDATION} keyword to control how SDEP pointers are to be validated.

You can also specify additional functions that are associated with the analysis function:

- Use the \texttt{RAP VALIDATION} keyword to control how RAPs are processed.
- Use the \texttt{LARGEST DATABASE RECORDS} keyword to specify the number of largest database records to be tracked by the analysis process.
- Use the \texttt{REPORT} and \texttt{THRESHOLD} subcommands to control the generation of analysis reports and exception testing.

The example in Figure 101 requests that analysis be performed on all output areas that are created by the \texttt{CHANGE} command. Direct pointers are validated by specifying \texttt{FULL} for the \texttt{POINTER VALIDATION} keyword. SDEP pointers (if any exist) are validated by using the same technique. For more information, see “Pointer validation” on page 97. Default reports are to be produced and no threshold checking is performed.

**Figure 101 Requesting analysis during DEDB change**

\begin{verbatim}
CHANGE  DBD=dbdname.
       POINTER VALIDATION=FULL
\end{verbatim}

**Making an image copy during DEDB change**

You can request that one or more image copy data sets be produced during the change function. Specify the \texttt{IC} subcommand for each image copy to be produced. For details about the \texttt{IC} subcommand, see Chapter 10, “Making an image copy.”
Enhancing performance

The example in Figure 102 requests the creation of a single output image copy data set for each area of the restructured DEDB.

**Figure 102  Requesting an image copy during DEDB change**

```
CHANGE DBD=dbdname,IAREA=ALL,OAREA=ALL
  IC DSNAME='dataset-name-mask',UNIT=TAPE,DISP=(NEW,CATLG)
```

Enhancing performance

The following methods might significantly improve the performance of the change function:

- caching
  - input caching
  - output caching
- thread processing
  - input threads
  - output threads

The effect of input and output caching on the change function depends on the following factors:

- type of input (image copy or area data set)
- type of caching that is used
- how SDEP segments (if any) are processed

Fast Path Offline Suite products include thread processing features that let you input and output DEDB areas in parallel for parallel processing. Processing multiple areas of a DEDB simultaneously can improve the performance of the products. The degree of performance enhancement that is provided by these features is dependent on the characteristics of your DEDB and the resources of your site.

The INPUT_THREADS keyword lets you specify the parallel input of a DEDB to the change function. The OUTPUT_THREADS keyword lets you specify parallel output to a DEDB from the change function.
Input caching

Input caching with the ICACHE keyword during the change function can reduce elapsed time for the function, at the cost of increased CPU utilization. A dataspace is used to implement the cache. If the change function uses an image copy as input, the function automatically uses caching of IOVF CIs. It also automatically caches SDEP CIs, if an SDEP segment is defined. If resources are not available to support the required IOVF and SDEP caching, an error message is generated and the change function terminates.

If the change function uses an area data set as input, input caching is optional. If resources are available, specify IOVF input caching. If the value LOGICAL is specified for the SDEP_PROCESS keyword, and resources will allow, specify SDEP caching. If the value NONE or PHYSICAL is specified for the SDEP_PROCESS keyword, caching of SDEP CIs is not beneficial and is not recommended.

Output caching

Output caching with the OCACHE keyword during the change function can reduce elapsed time for the function, at the cost of increased CPU utilization. A data space is used to implement the cache. If you specify output caching, specify it for both IOVF storage and SDEP storage. If you specify output caching and resources are not available to support the request, a warning message is generated and processing continues.

The change function cannot write a CI to the output area before it has been formatted. For example, if a UOW requires one or more IOVF CIs but formatting has not yet reached the IOVF portion, a delay in processing occurs while the intervening CIs are formatted. Output caching holds the output from the change function in the data space, eliminating the delay and allowing processing of the next UOW.

Input thread processing

The INPUT_THREADS keyword lets you specify the maximum number of input areas to process concurrently. If resources are not available to support the specified number, parallel processing occurs only on the number that resources can support.

The following considerations apply to using INPUT_THREADS as a means of significantly enhancing the change function:

- The default (maximum) value for INPUT_THREADS is determined automatically by the product, based on the number of input areas, CPU processors, and other system resources.
Specifying a number less than the automatic default might improve performance, but specifying a number greater than the default value is ignored.

Output thread processing

The OUTPUT_THREADS keyword specifies the maximum number of threads to use for parallel output. If resources are not available to support the specified number, parallel processing occurs only on the number that resources can support.

The following considerations apply to using OUTPUT_THREADS as a means of enhancing the change function:

- The parameter that is set by OUTPUT_THREADS specifies the maximum number of parallel threads to be output from the function. The default is the number of input threads in use (the same value that is specified on the INPUT_THREADS keyword).
- All output areas are opened simultaneously, and all output thread resources must be available throughout output processing. Sufficient resources must be available to accommodate the number specified.
- The number of OUTPUT_THREADS to specify depends on the number of output and input areas.
  - Same number of input areas and output areas
    - If the change function processes the same number of input and output areas, select a value equal to the parameter that is set by INPUT_THREADS, if resources allow.
    - If resources are not available to accommodate the number of input areas, specify the largest number that are available. Specifying a value greater than the number of input threads does not improve function performance because the extra resources are not used.
    - Ensure that adequate resources are available to support your selection.
  - Greater number of output areas than input areas
    - If the number of output areas of the change function is greater than the number of input areas, select a value equal to the number of areas that are output by the function, if resources allow.
Sample CHANGE command scenarios

- If resources are not available to accommodate the number of output areas, specify the largest number that resources can accommodate. Specifying a value greater than the number of output areas does not improve function performance because the extra resources are not used.

- Ensure that adequate resources are available to support your selection.

Sample CHANGE command scenarios

For scenarios that show how to use the CHANGE command with key related keywords and subcommands, see Appendix C, “Sample command scenarios.”
Unloading and Reloading a DEDB

This chapter provides information on the capabilities, setup, and use of the unload and reload functions that are provided by Fast Path Reorg/EP. These functions let you unload a DEDB into any of several file formats, and reload from any of the file formats to a DEDB. This chapter discusses the following topics:

Unloading and reload function overview .................................................. 169
Preparing for an unload and reload ......................................................... 170
Unload and reload function restrictions ................................................... 172
Unload function inputs and outputs ......................................................... 174
DBRC considerations ............................................................................. 175
UNLOAD command keywords and subcommands ................................ 175
Specifying input and output ................................................................. 176
  Selecting areas for input to the unload ................................................. 176
  Selecting areas for output .................................................................. 178
Allocating input area data sets ............................................................. 178
  Using DD statements ........................................................................ 178
  Using dynamic allocation .................................................................. 179
Allocating output unload data sets ....................................................... 179
  Using JCL to allocate the unload data set ......................................... 180
  Dynamic allocation for the unload data set ...................................... 180
  Using generation data set groups ....................................................... 181
Unloading a DEDB by using a generalized randomizer ....................... 182
  Unloading the entire database .......................................................... 183
  Limiting the number of output files ................................................. 183
  Creating a separate unload file for each area ..................................... 184
Controlling segment expansion ............................................................. 185
Controlling segment compression ......................................................... 185
Selecting SDEP processing method ...................................................... 186
Controlling subset pointer retention ...................................................... 187
Controlling the unload file format ........................................................ 188
Selecting database segments for output ............................................. 189
  Including and excluding segments for output ................................... 189
  Narrowing the selection .................................................................. 190
Changing segment contents ............................................................... 192
Processing considerations for randomization of root segments to unprocessed areas .......................... 194
Samples of INCLUDE, EXCLUDE, and OUTPUT subcommands .......................... 195
Controlling error tolerance ................................................................. 195
Making an image copy during an unload ............................................. 196
Enhancing performance ....................................................................... 197
  Unload function input caching .......................................................... 197
  Unload function input thread processing ......................................... 197
Reload function inputs and outputs ...................................................... 198
Reload DBRC considerations ............................................................... 199
RELOAD command keywords and subcommands ............................... 200
Specifying input and output .................................................................. 202
  Specifying areas to be read from unload input ............................... 202
  Specifying areas to be reloaded ....................................................... 203
Allocating input area data sets ............................................................ 204
Allocating output area data sets .......................................................... 205
  Using DD statements ................................................................. 205
  Using dynamic allocation to name the output area data set .......... 206
  Using the ALLOCATE subcommand to customize dynamically defined output areas ................................................................. 206
  ALLOCATE subcommand sample scenarios ..................................... 210
Reloading an area from multiple input files ......................................... 210
Controlling free space in IOVF ............................................................ 211
Placing segments with load control ...................................................... 211
  Defining segments for load control placement .............................. 213
  Placement control using segment content .................................... 214
  Placement control using segment counts ...................................... 216
Controlling segment expansion .......................................................... 217
Controlling segment compression ....................................................... 218
Selecting SDEP processing method ...................................................... 218
Selecting database segments for input ............................................... 219
  Including and excluding segments for input ............................... 219
  Narrowing the selection .............................................................. 220
Changing segment contents ............................................................... 223
Processing considerations for randomization of root segments to unprocessed areas ................................................................. 224
Handling exceptions .......................................................................... 225
  Assigning an exception limit ....................................................... 226
  Writing to a discarded data file ..................................................... 227
Rebuilding indexes during reload ....................................................... 229
Analyzing the DEDB during reload ..................................................... 229
Making an image copy during reload .................................................. 230
Enhancing performance ..................................................................... 231
  Reload function output caching .................................................... 231
  Reload function thread processing .............................................. 231
  Using SORT keyword with reload ............................................... 232
  Using the file sort utility .............................................................. 232
  Using SORT_OPTION keyword with reload ............................... 232
UNLOAD and RELOAD program extensions ....................................... 233
Sample UNLOAD and RELOAD command scenarios .......................... 233
Unload and reload function overview

Fast Path Reorg/EP unload and reload functions allow you to unload a DEDB into one of several file formats and to reload from any unload format to a DEDB.

- Use the UNLOAD command to unload one or more areas from a DEDB to formats that are compatible with any of the following:
  - IBM HD Reorganization Unload utility
  - IBM MSDB maintenance utility
  - IBM Fast Path Basic Tools for z/OS DEDB Reload utility
  - BMC Software TRIMAR FAST PATH UNLOAD/RELOAD product

- Use the RELOAD command to load one or more areas of a DEDB from input in any of the supported unload formats

The UNLOAD command can be used to produce an unload file to be processed later by the IBM HD Reorganization Reload Utility. For example, the file can be used to create an HDAM database for BTS testing. You can also use the unload/reload functions to convert between DEDBs and Main Storage Databases (MSDBs). For additional compatibility with legacy application programs, the UNLOAD command can be used to produce a TRIMAR unload format or IBM DBT unload format.

If you have created an index or indexes to the DEDB by using the BMC Fast Path Indexer/EP product, you can rebuild any and all indexes registered for the DEDB at the same time the reload function is performed.

When you need to modify the data content of a segment, you can accomplish it in two ways. The DEDB change function and the unload/reload functions allow segment data to be modified by using the OUTPUT subcommand. If more extensive changes are required, the UNLOAD command can be used to create an unload file. You can then use program extensions FABEUR7 and FABEUR6 to make changes to the data. Changed data is then used as input to the RELOAD command.

The DEDB change function restructures a DEDB by reading the DEDB, and writing the restructured DEDB directly to DASD. Because this method does not require separate unload and reload steps, it is the most efficient way to restructure a DEDB if sufficient DASD resources are available.

Because the Fast Path/EP reorganization function does not require separate unload and reload steps, it is the most efficient way to reorganize a DEDB to reclaim free space without making structure changes. It can also be used to extend the space that is allocated to the IOVF and SDEP portions of the area.
Preparing for an unload and reload

Several tasks are necessary to prepare for an unload and reload operation:

1. Modify the database definition (DBD) to incorporate DEDB changes.

   Use the DBDGEN and ACBGEN utilities to construct the ACB that contains the modified definition. Save a copy of the original ACB. The original ACB is required to access the existing area data sets during the unload function.

   Examples of DBD redefinition that could occur during database change using unload/reload processing include:
   - modifying area space parameters
   - increasing or decreasing the number of areas
   - changing a database structure
   - adding or deleting segments
   - adding or deleting subset pointers

   If these changes affect any indexes that are registered to the DEDB that were created using the BMC Software Fast Path Indexer/EP product, you must incorporate these changes into the index registration.

   **NOTE**
   If the original DBD contains a definition for an SDEP segment, you must select the technique to be used for processing these segments in the unloaded database.

2. Make image copies of areas before processing.

   BMC recommends that a valid image copy be made of each area before the area is processed by the unload function. This practice ensures that, if problems arise, you can perform adequate recovery of the original area. The unload file created as output from the unload function should not be used as an alternative to making an image copy. You can specify execution of the image copy function as part of the unload function or as a separate stand-alone process.

3. Decide the type of input to the unload function.

   You can use the original data sets or image copy data sets as input to the unload function:
   - If input to the unload function is the original area data sets, the unload function is a DASD-to-tape operation.
Preparing for an unload and reload

Chapter 6 Unloading and Reloading a DEDB

If input to the unload function is *image copy* data sets, the unload function is a tape-to-tape operation. This method requires sufficient cache resources to process the input data. For information about using caching with the unload function, see “Enhancing performance” on page 197. Regardless of whether an image copy is used as input to the unload function, BMC Software recommends that you make an image copy of the original areas before executing the unload function.

4. Make any required DBRC changes.

   Consider the implications that the changes to your DEDB will have in regard to DBRC. For example, if you are adding one or more areas to the database, you must add the new areas to the DBRC registrations. For more DBRC activity that is required before executing the reload function, see “Reload DBRC considerations” on page 199.

5. Make any required changes to the randomizer.

   Consider the implications that changes to your DEDB will have in regard to the randomizer. If you are adding one or more areas to the database, for example, you might need to modify the randomizer.

   The unload function uses the randomizer that is specified by the new DMB in the NEWACB DD statement to select the output file to which the records are written. The randomizer is obtained from the NEWRESLB DD statement, if supplied, or from STEPLIB. The randomizer specified by the old DMB in the IMSACB library is not used.

   The reload function uses the randomizer that is specified by the new DMB in the IMSACB library. The randomizer is obtained from the IMSRESLB DD statement, if supplied, or from STEPLIB. The randomizer that is specified by the old DMB is not used.

   **NOTE**

   BMC recommends using an area-specific randomizer because it offers better control over the segment distribution process.

6. Consider the implications of using a generalized randomizer.

   If you use a generalized randomizer and any changes are made to the DEDB that affect the number of root anchor points (RAPs), then all areas in the DEDB must be input to the unload process at the same time. For information about completing the unload process when a generalized randomizer is used, see “Unloading a DEDB by using a generalized randomizer” on page 182.
7. Consider suppression of repetitious messages.

For each anomaly encountered in the area, the unload or reload function generates a message with a specific suffix (severity) level. When the number of places that a particular condition exists is large, a large number of messages is produced. You can reduce the number of repetitious messages produced by using the MESSAGE_SUPPRESSION keyword to specify the maximum number of times that an informational, warning, error or critical message is to be produced. For the unload and reload processes, this keyword functions in the same manner as it does for the Fast Path/EP analysis process. For more information, see “Suppressing repetitious messages” on page 111.

8. Consider rebuilding indexes after processing.

If the changes to the DEDB require changes to its registered index (or indexes), you must rebuild the affected index (or indexes). You can specify execution of the index build function as part of the reload function or as a separate stand-alone process.

9. Consider analyzing areas after processing.

Analyzing the areas after the reload function can ensure that the results of your changes match your expectations. By default, a quick (checksum) analysis is executed as part of the reload command if you have a license for the Fast Path Analyzer/EP product. If a more detailed analysis or no analysis is desired, you can override the default by specifying the POINTER_VALIDATION keyword with the appropriate parameter value. You can specify execution of the analysis function as part of the reload function or as a separate stand-alone process.

10. Make image copies of areas after processing.

Making an image copy of the areas after the reload function and before any applications access the areas is required to ensure that if problems arise, you can perform adequate recovery. You can specify execution of the image copy function as part of the reload function or as a separate stand-alone process.

Unload and reload function restrictions

The following restrictions apply to the database unload and reload function:

- Only one UNLOAD or RELOAD command per job step can be specified, and it must be the only command in the PFPSYSIN input.

- When executing an UNLOAD process, the database must be offline. This process is not designed to execute against an online DEDB.
You cannot change segment names of root segments (except in databases consisting only of root segments). You can change other segment names.

You cannot change the hierarchical path of a dependent segment (cannot insert segments between existing segments).

You cannot change a direct dependent segment to an SDEP or an SDEP to a direct dependent segment.

If SMS-managed storage classes are used for initialized areas and the defined cluster spans the area across multiple volumes, you must allocate the area to a guaranteed space storage class. Fast Path Reorg/EP will not format out any portion of the area that resides on a candidate volume.

---

**NOTE**

If you use the CONFIGURE_AREA, CONFIGURE_IOVF, CONFIGURE_RAA or CONFIGURE_SDEP keyword with the ALLOCATE subcommand to dynamically define the output area for a RELOAD command process, do not allocate the area to a guaranteed space storage class. If you do so, the guaranteed space storage class will override the configuration specifications.

---

When a change is made to the key of a dependent segment with an unload or reload process, segments must remain in ascending key sequence under the root key. Fast Path Reorg/EP does not have a facility to resequence the dependent segments. A failure of the reload process occurs if the change to a dependent segment key is not specified in ascending key sequence.

If an image copy is used as input to the database unload function, an IBM MVS™ dataspace is used. The size of the required dataspace cannot be greater than 2 GB. If this limit is exceeded, Fast Path Reorg/EP issues an error message and terminates.

---

**NOTE**

The 2 GB limit applies to the combined size of the IOVF and SDEP portion of the database, not the size of the image copy. Fast Path Reorg/EP tries to minimize the size of the required dataspace by using an internal compression algorithm on the IOVF and SDEP blocks.

---

When using SDEP_PROCESS=PHYSICAL on the UNLOAD command, you cannot perform any of the following tasks:

— change the CI size of the area

— change the compression parameters for the SDEP segment

— specify the OUTPUT subcommand for the SDEP segment
— use the FABEUR6 and FABEUR7 program extensions to process the unload file
— modify the randomization parameters such that root segments are randomized to a different area
— create a single unload file from multiple areas (you must create an unload file for each area)

Unload function inputs and outputs

As shown in Figure 103, the unload function processes input from an area data set or an image copy data set. Use the unload function to unload one or more areas from a DEDB in any of the available formats. For more information about available formats, see “Controlling the unload file format” on page 188.

The ACB of the database that is being unloaded is used to describe the input areas and is referenced by the IMSACB DD statement. The ACB of the database to be reloaded is referenced by the NEWACB DD statement and is used to describe the output files.

Figure 103 Unload function inputs and outputs

The input to the unload function is the areas being unloaded. The input can be an area data set or an input image copy data set.

The output from the unload function is the unloaded output areas.
The Input Area Summary Report, Output Area Summary Report, and the Database Summary Report are generated by the unload function to summarize unload activity. For an example of these reports, see the PFPUNLD member in the REPORTS data set.

**DBRC considerations**

When DBRC is active during execution of the unload function, and the area is registered, the authorization level of the area data set is requested as shown in Table 12. If the area data set is used as input, the data set name must match the name that is registered with DBRC.

<table>
<thead>
<tr>
<th>Command</th>
<th>Authorization</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNLOAD (area data set input)</td>
<td>Read-with-integrity (RD)</td>
</tr>
<tr>
<td>UNLOAD (image copy inputs)</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Multiple area data sets (MADS) are not supported. The unload function searches the ADS (area data set) list that is registered for each area (in collating sequence by ddname). The product selects the first ADS that is marked as available for use and that has no error queue elements (EQEs). If an ADS meets both of these criteria, it is the only ADS that is unloaded. All other area data sets are ignored.

If no area data set is marked as available, or if all available area data sets contain one or more EQEs, the unload is not performed for that area.

**UNLOAD command keywords and subcommands**

Table 13 lists the keywords and subcommands that are available for the UNLOAD command.

<table>
<thead>
<tr>
<th>Function</th>
<th>Command or subcommand</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>selecting the database and areas</td>
<td>UNLOAD</td>
<td>DBD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IAREA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OAREA</td>
</tr>
</tbody>
</table>
Specifying input and output

The DBD keyword identifies the name of the DEDB (DBD name) to be unloaded.

Selecting areas for input to the unload

The IAREA keyword identifies the names of the areas to be read from the database that is described in the IMSACB library. You can specify one or more area names. If you omit the IAREA keyword, all areas that are defined in the DEDB are read.

Table 13 UNLOAD command keywords and subcommands (part 2 of 2)

<table>
<thead>
<tr>
<th>Function</th>
<th>Command or subcommand</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>allocating the area data sets</td>
<td>UNLOAD</td>
<td>INPUT_DSN_MASK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OUTPUT_DSN_MASK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OFILECTL</td>
</tr>
<tr>
<td>controlling segment expansion</td>
<td>UNLOAD</td>
<td>EXPAND</td>
</tr>
<tr>
<td>controlling segment compression</td>
<td>UNLOAD</td>
<td>COMPRESS</td>
</tr>
<tr>
<td>selecting SDEP processing method</td>
<td>UNLOAD</td>
<td>SDEP_PROCESS</td>
</tr>
<tr>
<td>controlling subset pointer retention</td>
<td>UNLOAD</td>
<td>SUBSET_POINTERS</td>
</tr>
<tr>
<td>controlling the unload file format</td>
<td>UNLOAD</td>
<td>FORMAT</td>
</tr>
<tr>
<td>selecting database segments for output</td>
<td>INCLUDE</td>
<td>SAMPLE_INTERVAL</td>
</tr>
<tr>
<td></td>
<td>EXCLUDE</td>
<td>SAMPLE_LIMIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SEGMENT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WHERE</td>
</tr>
<tr>
<td>changing segment contents</td>
<td>OUTPUT</td>
<td>all associated keywords</td>
</tr>
<tr>
<td>controlling error tolerance</td>
<td>UNLOAD</td>
<td>ERROR_THRESHOLD</td>
</tr>
<tr>
<td>making an image copy while unloading</td>
<td>IC</td>
<td>all associated keywords</td>
</tr>
<tr>
<td>enhancing performance</td>
<td>UNLOAD</td>
<td>INPUT_THREADS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ICACHE</td>
</tr>
<tr>
<td>controlling processing of root segments to</td>
<td>UNLOAD</td>
<td>BYPASS_RECORD</td>
</tr>
<tr>
<td>non-processed areas</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For more information about the following topics, see the Fast Path/EP Series Reference Manual:

- syntax of commands, subcommands, and keywords that are discussed in this book
- diagrams that show the syntax and available parameters and values for Fast Path Offline Suite commands and subcommands

Specifying input and output

The DBD keyword identifies the name of the DEDB (DBD name) to be unloaded.

Selecting areas for input to the unload

The IAREA keyword identifies the names of the areas to be read from the database that is described in the IMSACB library. You can specify one or more area names. If you omit the IAREA keyword, all areas that are defined in the DEDB are read.
Multiple areas can be specified on the IAREA keyword by using any combination of area names, area numbers, or a area ranges. The following parameters are available for the IAREA keyword:

- **IAREA=ALL** (default) or **IAREA=*** specifies all areas of the DEDB.

- **IAREA=areaname** specifies one or more areas by using the one-character to eight-character area name for each area specified. Multiple area names must be enclosed in parentheses and separated by commas.

- **IAREA=areanumber** specifies one or more areas by using the one-character to five-character area number for each area specified. Multiple area numbers must be enclosed in parentheses and separated by commas.

- **IAREA=(RANGE=(startarea,endarea))** specifies a consecutive range of areas using either areaname or areanumber parameters. The area associated with **startarea** must be less than the area number associated with **endarea**.

An asterisk (*) can be used to specify all areas of the DEDB. When the * character is used with the RANGE keyword, it can be used to specify the beginning or ending range for specific areas of the DEDB.

To unload an entire DEDB, use a command set like the example shown in **Figure 104**.

**Figure 104  Sample control statement for unloading all areas**

```
UNLOAD DBD=dbname, IAREA=ALL, OAREA=ALL
```

To unload specific areas, use a command set like the example shown in **Figure 105**.

**Figure 105  Sample control statement for unloading specific areas**

```
UNLOAD DBD=dbname, IAREA=(AREANAM1,AREANAM3,RANGE=(5,8)),
OAREA=(AREANAM1,AREANAM3,RANGE=(5,8))
```

This control statement uses a combination of area names and area numbers to request that the specified areas are to be input to the unload process. The areas named **AREANAM1** and **AREANAM3** will be input. Also, all consecutive areas from area number 5 to area number 8 (**area5, area6, area7** and **area8**) will be input to the unload process.
Selecting areas for output

The OAREA keyword identifies the names of the areas to be written from the database that is described in the NEWACB library. You can specify one or more area names. If you omit the OAREA keyword, all areas that are defined in the DEDB are written to the unload file.

Multiple areas can be specified on the OAREA keyword by using any combination of area names, area numbers, or a area ranges. The following parameters are available for the OAREA keyword:

- OAREA=ALL (default) or OAREA=\* specifies all areas of the DEDB.
- OAREA=\areaname\ specifies one or more areas by using the one-character to eight-character area name for each area specified. Multiple area names must be enclosed in parentheses and separated by commas.
- OAREA=\areanumber\ specifies one or more areas by using the one-character to five-character area number for each area specified. Multiple area numbers must be enclosed in parentheses and separated by commas.
- OAREA=(RANGE=(\startarea,\endarea\)) specifies a consecutive range of areas using either \areaname\ or \areanumber\ parameters. The area associated with \startarea\ must be less than the area number associated with \endarea\.

An asterisk (\*) can be used to specify all areas of the DEDB. When the * character is used with the RANGE keyword, it can be used to specify the beginning or ending range for specific areas of the DEDB.

Values for the OAREA keyword function in the same manner as for the IAREA keyword. See the examples shown for the IAREA keyword in Figure 104 and Figure 105.

Allocating input area data sets

The areaname DD statement identifies the area data set to be read. The areaname DD statement can specify an image copy data set. If the areaname DD statement is omitted from the JCL, Fast Path Reorg/EP attempts to dynamically allocate it.

Using DD statements

The IMSACB DD statement identifies the ACB library containing the database definition that describes the area referenced by the areaname DD statement.
Using dynamic allocation

If you are using dynamic allocation, do not include the areaname DD statement. Fast Path Reorg/EP tries to obtain the data set name for allocation in the following order:

1. If the INPUT_DSN_MASK keyword is specified, it is used to generate the data set name. The data set name can specify an image copy.

2. If DBRC is active and the area is registered, the registered area data set name is obtained from DBRC.

3. The STEPLIB is searched for the DFSMDA member that contains the data set name for this area.

If DBRC is active, the area is registered with DBRC, and the data set to be unloaded is not an image copy, the allocated data set name must match the registered data set name, regardless of how it is allocated.

In the example shown in Figure 106, the input area data sets are dynamically allocated.

Figure 106 UNLOAD command with dynamic allocation for input

```
UNLOAD DBD=dbname,
    IAREA=ALL,INPUT_DSN_MASK='PFP.&DBD.&AREA'.
```

Allocating output unload data sets

The unload data set can be allocated by using DD statements that are supplied in the JCL, or by using dynamic allocation. BMC Software recommends dynamic allocation because of the flexibility and improved control that it provides.

You can specify the optional OFILECTL subcommand to combine the data targeted to one or more output areas into a single output data set. One unload data set is produced for each OFILECTL subcommand that is specified. Do not code the OAREA keyword on the UNLOAD command. Code this keyword on the OFILECTL subcommand to specify the output areas to be placed into the unload data set.

If no OFILECTL subcommand is specified, a separate unload data set is produced for each output area. The DD statement for each unload data set is named OAREAxxx, which uses a three-digit area number, or OARxxxx, which uses a five-digit area number. The OAREAxxx DD statement can only be used for area numbers 1 through 999. The OARxxxx DD statement can be used for area numbers 1 through 2048, and
must be used for area numbers greater than 999. If this DD statement is omitted from
the JCL, Fast Path Reorg/EP tries to dynamically allocate the output unload data set
by using the OUTPUT_DSN_MASK keyword. The data set name that is generated by
using this mask must be a preexisting cataloged data set.

You can specify the PFPTOTAL DD statement to allocate an optional output
PFPTOTAL data set. This output data set can be used as an audit file to verify
segment output totals after using the unload process.

The PFPTOTAL data set is designed for input to a user-written program. For more
information about the PFPTOTAL DD statement and the record format, see the Fast

Using JCL to allocate the unload data set

Use the DDNAME keyword to indicate that the unload data set is allocated by using
JCL DD statements.

The example in Figure 107 uses a JCL DD statement to allocate an unload data set. All
output areas are written into a single unload data set.

Figure 107  JCL to allocate an unload data set

```plaintext
//PFP EXEC PGM=PFPMAIN,REGION=0M
//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
// ddname1 DD UNIT=3400,DISP=(NEW,CATLG),
//      DSN=your.dataset.name
//PFPSYSIN DD *
UNLOAD DBD=dbname,AREA=ALL
OFILECTL OAREA=ALL,DDNAME=ddname1
/*
```

Dynamic allocation for the unload data set

In the example shown in Figure 108, the input and output area data sets are
dynamically allocated. The output data set names are in the form
PFP.dbdname.ULxxxx, where xxxx is the five-digit area number.

Figure 108  UNLOAD command with dynamic allocation for output

```plaintext
UNLOAD DBD=dbname.
    OAREA=ALL,OUTPUT_DSN_MASK='PFP.&DBD.UL&AREA#5'
```
You can specify the optional OFILECTL subcommand to combine the data targeted to one or more output areas into a single output data set. One unload data set is produced for each OFILECTL subcommand that is specified. Do not code the OAREA keyword on the UNLOAD command. Instead, code this keyword on the OFILECTL subcommand to specify the output areas to be placed into the unload data set.

Use the DSNAME keyword to indicate that the unload data set is to be dynamically allocated, and to supply a mask for constructing the data set name. Substitution variables within the mask can be used to generate unique names for each unload data set.

Additional keywords such as UNIT, DISP, and DATACLAS can be used to control the allocation and disposition of the data set, similar to the corresponding JCL keywords.

The example in Figure 109 requests the creation of two unload data sets. The first data set contains data for output areas areanam1 and areanam2. The second data set is allocated to a disk device and contains data for output area area1024.

**Figure 109  Dynamic allocation of unload data sets**

```
UNLOAD DBD=dbdname, IAREA=ALL
  OFILECTL OAREA=(areanam1, areanam2),
  DSNAME='dataset-name1-mask',
  UNIT=TAPE, DISP=(NEW,CATLG)
  OFILECTL OAREA=(area1024)
  DSNAME='dataset-name2-mask'
  UNIT=3390, VOLSER=123456,
  SPACE=(CYL,100,10,RLSE),
  DISP=(NEW,CATLG,DELETE)
```

**NOTE**
The ddname that is used for dynamic allocation of the output data sets depends on the area number of the first area that is listed in the OAREA keyword. If the first area that is listed in the OAREA keyword has an area number between 1 and 999, the OAREAxxx DD is used for dynamic allocation, where xxx is the 3-digit area number. If the first area that is listed in the OAREA keyword has an area number between 1000 and 2048, OARxxxxx is used for dynamic allocation, where xxxxx is the area number. If an OAREAxxx or OARxxxxx DD statement is supplied in the JCL, it is used instead of the dynamic allocation parameters.

**Using generation data set groups**
The data set name specified with the DSNAME keyword can refer to a generation data group (GDG). Code the DSNAME keyword to specify the base name for the GDG. As an option, you can include the “(+1)” relative generation number as part of the DSNAME keyword.
The example in **Figure 110** requests that all unload data be written to a single data set. The unload data set is allocated as the next generation within a predefined GDG.

**Figure 110  Using a generation data set group**

<table>
<thead>
<tr>
<th>UNLOAD DBD=dbdname,AREA=ALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFILECTL OAREA=ALL,DSNAME='gdg-dataset-name-mask(+1)',</td>
</tr>
<tr>
<td>DISP=(NEW,CATLG),UNIT=TAPE</td>
</tr>
</tbody>
</table>

**Unloading a DEDB by using a generalized randomizer**

If you use a generalized randomizer with a DEDB, any change in the number of root anchor points (RAPs) affects the distribution of records across the DEDB. Changes that affect the number of RAPs include the following:

- changes to the UOW or ROOT (change in RAA value)
- increasing or decreasing the total number of areas in the DEDB

These changes to the number of RAPs require that all areas in the DEDB be input to the unload process at the same time, preferably in the same command set.

The design flexibility of Fast Path Reorg/EP enables parallel processing of DEDBs in multiple ways. The product will not terminate processing if all areas in the DEDB are not defined to the unload process when a generalized randomizer is used.

**WARNING**

Failure to include all existing areas as input to the unload process at the same time can cause records to be discarded because of re-randomization by the generalized randomizer. In this situation, Fast Path Reorg/EP issues message BMC111178W to indicate that it found a change in the structure of the database that affects the entire DEDB but not all areas were included in the process. For example, if the unload is processing one area at a time and a record moves from *area1* to *area2* because of re-randomization and there is no output file for *area2* (no OAREA002 DD statement present in the JCL), the entire record is discarded.

Three different techniques are available to complete the unload process when a generalized randomizer is used.
Unloading the entire database

The simplest method to unload a DEDB when a generalized randomizer is used is to execute the unload with only the DBD keyword (without IAREA and OAREA keywords) as shown in Figure 111. By default, all areas are processed by the UNLOAD command. Input areas are dynamically allocated.

Figure 111 Unloading entire area when using generalized randomizer

```
//PFP EXEC PGM=PFPMAIN,REGION=0M
//STEPLIB DD DSN=BMC.PFP LOAD,DISP=SHR
// DD DSN=IMSVS. RESLIB,DISP=SHR
// NEWACB DD DSN=IMSVS.NEW.ACBLIB,DISP=SHR
// IMSACB DD DSN=IMSVS.OLD.ACBLIB,DISP=SHR
//OAREA001 DD DSN=PFP.UNLOAD.AREA1,DISP=(,CATLG)
// DD SYSDA,SPACE=(CYL,(25,5),RLSE)
//OAREA002 DD DSN=PFP.UNLOAD.AREA2,DISP=(,CATLG)
// DD SYSDA,SPACE=(CYL,(25,5),RLSE)
//OAREA003 DD DSN=PFP.UNLOAD.AREA3,DISP=(,CATLG)
// DD SYSDA,SPACE=(CYL,(25,5),RLSE)
//OAREA004 DD DSN=PFP.UNLOAD.AREA4,DISP=(,CATLG)
// DD SYSDA,SPACE=(CYL,(25,5),RLSE)
//PFPPRINT DD SYSOUT=* 
//PFPSYSIN DD * 
UNLOAD DBD=DB400
```

Limiting the number of output files

If you have storage constraints (DASD or tape units) and want to limit the number of output files, specify the OFILECTL subcommand on the UNLOAD command as shown in Figure 112. Specify IAREA=ALL, or omit the IAREA keyword which will default to ALL. Specifying IAREA=ALL causes all areas to be read as input. Use the OFILECTL subcommand to specify the output file data set name, but do not specify the OAREA keyword to ensure that the data for all areas will be output to a single file. Input areas are dynamically allocated.

Figure 112 Specifying one output file when using generalized randomizer

```
//PFP EXEC PGM=PFPMAIN,REGION=0M
//STEPLIB DD DSN=BMC.PFP LOAD,DISP=SHR
// DD DSN=IMSVS. RESLIB,DISP=SHR
// NEWACB DD DSN=IMSVS.NEW.ACBLIB,DISP=SHR
// IMSACB DD DSN=IMSVS.OLD.ACBLIB,DISP=SHR
//PFPPRINT DD SYSOUT=* 
//PFPSYSIN DD * 
UNLOAD DBD=DB400
OFILECTL DSNAMES='PFP.UNLOAD.DB400',DISP=(NEW,CATLG),
UNITY=SYSDA,SPACE=(CYL.45,15,RLSE)
```
Creating a separate unload file for each area

To create more than one output file, specify multiple OFILECTL subcommands, each with an OAREA keyword that specifies the desired output area or areas as shown in Figure 113. You must specify unique area names on each OAREA keyword. Do not specify the same output area name on more than one OFILECTL subcommand.

Figure 113  Specifying multiple output files when using generalized randomizer

```
UNLOAD DBD=DB400,IAREA=ALL
OFILECTL OAREA=(AREA1,AREA3)
    DSNAME='PFP.UNLOAD.DB400A',DISP=(NEW,CATLG),
    UNIT=SYSDA,SPACE=(CYL,25,15,RLSE)
OFILECTL OAREA=(AREA2,AREA4)
    DSNAME='PFP.UNLOAD.DB400B',DISP=(NEW,CATLG),
    UNIT=SYSDA,SPACE=(CYL,25,15,RLSE)
```

Creating a separate unload file for each area

To create a separate unload file for the data that is contained in each area, specify the UNLOAD command with the OAREA keyword that specifies the area name as shown in Figure 114. Repeat this process for each area in the DEDB. Because the input areas can be shared, all jobs can be executed concurrently.

**NOTE**

To select all areas for input, specify IAREA=ALL, or omit the IAREA keyword which will default to ALL.

Figure 114  Creating a separate unload file per area when using generalized randomizer

```
//PFP   EXEC  PGM=PFPMAIN,REGION=0M
//STEPLIB DD DSN=BMC.PFP.LOAD,DISP=SHR
//                       DD DSN=IMSVS.RESLIB,DISP=SHR
//NEWACB DD DSN=IMSVS.NEW.ACBLIB,DISP=SHR
//IMSACB DD DSN=IMSVS.OLD.ACBLIB,DISP=SHR
//AREA1  DD DSN=PFP.DB400.AREA1,DISP=SHR
//AREA2  DD DSN=PFP.DB400.AREA2,DISP=SHR
//AREA3  DD DSN=PFP.DB400.AREA3,DISP=SHR
//AREA4  DD DSN=PFP.DB400.AREA4,DISP=SHR
//OAREA001 DD DSN=PFP.DB400.UNLOAD,DISP=(,CATLG),
             UNIT=SYSDA,SPACE=(CYL,(15,5),RLSE)
//PFPPRINT DD SYSOUT=*  
//PPFSYSIN DD *  
UNLOAD DBD=DB400,OAREA=AREA1
```
Controlling segment expansion

As an option, you can expand compressed segment data as it is written to the unload file. This option is determined by the EXPAND keyword:

- EXPAND=NO (the default) causes segment data to be written to the unload file in the same format as it appears in the input area.
- EXPAND=YES causes compressed segment data to be expanded before it is written to the unload file.
- EXPAND=(segment1, segment2, ..., segmentn) causes only the compressed segments which are listed by name to be expanded before they are written to the unload file.

Figure 115 shows an example of the EXPAND keyword with the UNLOAD command.

Figure 115 Using the EXPAND keyword with unload

| UNLOAD DBD=dbdname, IAREA=ALL, OAREA=ALL, EXPAND=YES |

Figure 116 shows an example of using the EXPAND keyword to selectively specify segment expansion. Only segment1 and segment3 in area1 will be expanded; all other segments will be written to the unload file unchanged.

Figure 116 Selectively specifying segments for expansion with unload

| UNLOAD DBD=dbdname, IAREA=area1, OAREA=ALL, EXPAND=(segment1,segment3) |

Controlling segment compression

As an option, you can compress expanded segment data during the unload function. This option is determined by the COMPRESS keyword.

- COMPRESS=NO (the default) causes segment data to be written to the unload file in the same form as it appears in the input area.
- COMPRESS=YES causes expanded segment data to be compressed before it is written to the unload file.
- COMPRESS=(parameter1, parameter2, ..., parametern) causes only the expanded segments which are listed by name to be compressed before they are written to the unload file.
Selecting SDEP processing method

In the example shown in Figure 117, IAREA and OAREA default to ALL. The segments that have a compression exit specified in the new DBD will be compressed.

Figure 117  Using the COMPRESS keyword with unload

UNLOAD DBD=dbdname,COMPRESS=YES

Figure 118 shows an example of using the COMPRESS keyword to selectively specify segment compression for specified area. Only segment1, segment4 and segment5 in area1 will be compressed, all other segments will be written to the output area in the same format that they appear in the input area.

Figure 118  Selectively specifying segments for compression with unload

UNLOAD DBD=dbdname,IAREA=area1,
OAREA=ALL,COMPRESS=(segment1,segment4,segment5)

Selecting SDEP processing method

If the input database has an SDEP segment defined, you must indicate how these segments are to be processed. Multiple methods are available for processing SDEP segments during execution of the unload function. The option used is determined by the SDEP_PROCESS keyword:

- SDEP_PROCESS=LOGICAL causes SDEP segments to be processed in conjunction with the root segment that owns them. The SDEP segments in each output area are physically reordered into the same sequence as the parent root segments. The SDEP segments retain the same logical order (entry sequence) within each database record.

--- WARNING ---

If your application relies on the marker segment concept, do not use SDEP_PROCESS=LOGICAL.

---

- SDEP_PROCESS=PHYSICAL causes SDEP segments to be processed in the physical sequence in which they appear in the input area. They are inserted into the output areas in the same relative physical location, fully supporting the marker segment concept. They also retain the same logical order (entry sequence) within each database record, including the same cycle count and timestamp.
Controlling subset pointer retention

As an option, you can retain the settings for subset pointers during the unload function. This option is determined by the SUBSET_POINTERS keyword. If subset pointers are defined in the DEDB to be unloaded, you must specify a value for the SUBSET_POINTERS keyword:

- SUBSET_POINTERS=NO causes all subset pointers in the output areas to be cleared. Subset pointers from the input areas are not transferred.

### NOTE

The following restrictions and considerations apply when SDEP_PROCESS=PHYSICAL is specified on the UNLOAD command:

- You cannot reduce the CI size of the area; however, you can increase the CI size.
- You cannot change the compression parameters for the SDEP segment.
- You cannot specify the OUTPUT subcommand for the SDEP segment.
- You cannot use the FABEUR6 and FABEUR7 program extensions to process the unload file.
- You cannot modify the randomization parameters such that root segments will be randomized to a different area.
- You cannot create a single unload file from multiple areas (you must create an unload file for each area).
- SDEP pointer errors will not be detected. Consequently, any SDEP errors will not be applied to the error detection count value that you specify on the ERROR_THRESHOLD keyword.

- SDEP_PROCESS=NONE causes SDEP segments present in the input areas to be ignored; these segments are not placed in the output areas. If SDEP segment types are detected in the input area, a warning message is generated to indicate that SDEP segments will be removed.

Figure 119 shows an example of the SDEP_PROCESS keyword with the UNLOAD command.

**Figure 119 Using the SDEP_PROCESS keyword**

```
UNLOAD DBD=dbdname,SDEP_PROCESS=PHYSICAL
```
Controlling the unload file format

- SUBSET_POINTERS=YES causes the values of subset pointers from the input areas to be transferred to the output areas. The actual pointer value (RBA) is translated so that it refers to the new location of the segment after it has been placed into the output area.

Figure 120 shows an example of the SUBSET_POINTERS keyword with the UNLOAD command.

![Figure 120 Using the SUBSET_POINTERS keyword](image)

UNLOAD DBD=dbname,SUBSET_POINTERS=YES

### Controlling the unload file format

You can specify one of four formats for the output file created by the unload function. The format of the file is controlled using the FORMAT keyword:

- FORMAT=HDUNLOAD (the default) specifies the file to be written in a format compatible with the IBM HD Reorganization Unload Utility. This option allows the file to be used as input to the Fast Path Reorg/EP reload function. It can also be used as input to the BMC LOADPLUS product or the IBM HD Reorganization Reload utility, which provides an easy-to-use method for converting a DEDB to an HDAM database.

---

**WARNING**

If you specify SDEP_PROCESS=PHYSICAL on the UNLOAD command, HDUNLOAD is the only format that you can select for the FORMAT keyword. Processing of the UNLOAD will terminate if any format other than UNLOAD is specified on the FORMAT keyword when SDEP_PROCESS=PHYSICAL is specified.

---

- FORMAT=MSDBINIT specifies the file to be written in a format that is compatible with the IBM MSDB Maintenance Utility. Only root segments will be placed in the output file. This format is useful for converting a DEDB to a nonterminal-related MSDB.

- FORMAT=TFMT specifies the file to be written in a format that is compatible with the TRIMAR FAST PATH UNLOAD/RELOAD product.

- FORMAT=DBT specifies the file to be written in a format that is compatible with the DEDB Reload Utility component of the IBM Fast Path Basic Tools for OS/390®. For the record layout that is used when TFMT or DBT is specified as the unload format, see the *Fast Path/EP Series Reference Manual*. 

---

188  Fast Path Offline Suite User Guide
Selecting database segments for output

The segments that are selected during input processing are forwarded to output processing for record formatting. By default, all root segments and their dependent segments are selected, excluding SDEP segments. You can modify the selection of segments by using the optional INCLUDE and EXCLUDE subcommands. You can also use several keywords with the INCLUDE and EXCLUDE subcommands to further refine the selection process.

NOTE
To control SDEP segment selection, use the SDEP_PROCESS keyword discussed on page 186.

Including and excluding segments for output

The INCLUDE subcommand lets you specify criteria for the selection of segments that will be included in the unload file. If the INCLUDE subcommand selection criteria are met, the selected segments and their dependents are written to the unload file.

The EXCLUDE subcommand lets you specify criteria for selection of segments that will be omitted from the unload file. If the EXCLUDE subcommand selection criteria are met, the selected segments and their dependents are not written to the unload file.

NOTE
You cannot include a dependent segment and exclude its parent segment. Specifying a parent segment on the EXCLUDE subcommand also excludes all dependent segments.

The SEGMENT keyword specifies the name of the segment to which the selection criteria are applied. Additional selection criteria can be specified by using the WHERE keyword, which uses field names or positional identifiers with relational or Boolean logical expressions.

A simple INCLUDE or EXCLUDE subcommand uses the required SEGMENT keyword without additional keywords. Understanding the rules that apply to simple INCLUDE and EXCLUDE subcommands is prerequisite to using additional keywords with INCLUDE and EXCLUDE:

- By default, if no INCLUDE or EXCLUDE statement is used with the UNLOAD command, all roots and dependents are selected for processing.
Narrowing the selection

- The INCLUDE subcommand selects the segment that is identified by the SEGMENT keyword and all dependents of the segment. Including a segment in this manner also selects the parent hierarchy chain above the segment.

- The EXCLUDE subcommand excludes the segment that is identified by the SEGMENT keyword and all dependents of the segment. You cannot include a segment and exclude a root or a parent of the segment. Excluding a parent excludes the entire hierarchy beneath the parent.

A basic UNLOAD command with an INCLUDE and EXCLUDE subcommand is shown in Figure 121.

Figure 121  Using the UNLOAD command with INCLUDE and EXCLUDE subcommands

```
UNLOAD DBD=dbdname, IAREA=(areanam1,areanam3)
OAREA=(areanam1,areanam3)
INCLUDE SEGMENT=segname1
EXCLUDE SEGMENT=segname2
```

Narrowing the selection

While identifying the segments of interest for the unload using the INCLUDE and EXCLUDE subcommands, you can apply additional criteria to narrow the selection by using the following keywords:

- The WHERE keyword selects segments by applying conditional criteria that is specified in a Boolean expression.

- The SAMPLE_INTERVAL keyword selects segments by intervals of occurrence.

- The SAMPLE_LIMIT keyword sets a limit on the number of segments meeting the criteria that will be output to the unload file.

Conditional criteria

You can narrow segment selection by using the WHERE keyword. This combination applies relational and Boolean operators to apply conditional criteria when selecting segments. For information about the operations and syntax of the WHERE keyword and syntax rules for coding these expressions, see the Fast Path/EP Series Reference Manual.

Figure 122 shows a typical DEDB segment hierarchy. This hierarchy is used in subsequent examples in this section that show how the INCLUDE and EXCLUDE subcommands can be used with their associated keywords.
The example in Figure 123 is used to create a test system database from a production version by selecting a specific set of records. The example selects only the SEGA segments that have the field ROOTKEY equal to BMCPFP. For the selected SEGA segments, only SEGB (and its dependents) where position 23 is equal to 22 will be selected. Only SEGE segments (of the selected SEGA segments) created after the year 1997 are selected. Although SEGG is defined in the DBD, it is excluded from the unload file.

**Criteria that are applied to alternate segments**

You can specify segment selection, based on criteria that are applied to alternate segments, by using a segment-qualified field reference within the WHERE keyword. Only the parentage segments within the hierarchical path from the root can be used for qualification.

In the example in Figure 124, SEGC is selected only if FIELD1 in SEGB is equal to the character string BMC.
**Segment sampling**

The SAMPLE_INTERVAL keyword can be used to specify segment selection based on the frequency of the segment occurrence. For example, if you have specified SAMPLE_INTERVAL=5, every fifth segment is selected. For a root segment, every fifth occurrence in the database and all of its dependents will be selected. For a dependent segment, every fifth occurrence under its parent is selected.

The SAMPLE_LIMIT keyword can be used to specify the maximum number of segment occurrences to be selected. For example, if you have specified SAMPLE_LIMIT=500, the first 500 segments will be selected. For a root segment, the first 500 occurrences and all its dependents are selected. For a dependent segment, the first 500 occurrences within its parent are selected.

In the example in Figure 125, all areas are included in the process because the IAREA and OAREA keywords are not specified. The result of the INCLUDE subcommand is that every ninth occurrence of SEGA (and all its dependents) are written to the unload file, with a maximum of 5000 occurrences of SEGA. The EXCLUDE subcommand changes the result of the previous INCLUDE for dependent segment SEGE. For each selected SEGA segment, the first five segment occurrences for segment SEGE are omitted. If, for example, segment SEGE has eight occurrences, the last three segment occurrences are written to the unload file. If segment SEGE has only four occurrences under another selected SEGA, no SEGE segments (nor their dependents) are written to the unload file.

**Figure 125  UNLOAD command with SAMPLE_INTERVAL and SAMPLE_LIMIT keywords**

```
UNLOAD DBD=dbdname,
   INCLUDE SEGMENT=SEGA,
       SAMPLE_INTERVAL=9,
       SAMPLE_LIMIT=5000
   EXCLUDE SEGMENT=SEGE,
       SAMPLE_LIMIT=5
```

**Changing segment contents**

The OUTPUT subcommand and its associated keywords can be used to change the content of the segment. By default, the unload file will contain the full, unmodified contents of the input segments.
To modify the segment contents, you can use the OUTPUT subcommand with the FIELDS keyword. The FIELDS keyword specifies a list of expressions; the value of each expression is evaluated and placed into the output segment in the order that the expressions are specified. The length of the segment is computed by the product as the sum of the lengths of the values that are produced from these expressions. For information about the operations and syntax of the FIELDS keyword and syntax rules for coding expressions, see the Fast Path/EP Series Reference Manual.

**WARNING**

When the FIELDS keyword is used with the OUTPUT subcommand to change the key of a dependent segment with an unload process, the segments must remain in ascending key sequence under their parent. Fast Path Reorg/EP does not have a facility to resequence the dependent segments. The reload process terminates if the dependent segments are not maintained in ascending key sequence.

**NOTE**

When the FIELDS keyword is used with the OUTPUT subcommand to modify the content of a compressed segment, you must expand the segment by using the EXPAND=YES value or the EXPAND=(segment1,segment2,...,segmentn) syntax on the UNLOAD command. This expansion will ensure that the operands which are specified on the FIELDS keyword correspond to the proper segment columns.

Figure 126 shows an example where a literal value is being inserted into the output segment during a DEDB unload.

**Figure 126 Modifying output segment contents**

```
UNLOAD DBD=dbname,
    OUTPUT SEGMENT=SEGD
    FIELDS=(3:20,4P'0',23:*)
```

The content of the output segment can be controlled by using conditional qualification that is specified by the optional WHERE keyword on the OUTPUT subcommand. If multiple OUTPUT subcommands are specified for the same segment, the first subcommand in which the selection criteria is satisfied is used. If the selection criteria is not satisfied for any OUTPUT subcommand for a segment, the input segment is copied to the output area without modification.

Figure 127 shows an example where the content of a segment is conditionally modified during a DEDB unload. If the WHERE condition that is specified on the OUTPUT subcommand is met, the FIELDS keyword is used to modify the segment content. If the condition is not met, the segment content is unchanged.

**Figure 127 Using WHERE keyword to conditionally specify segment content (part 1 of 2)**

```
UNLOAD DBD=dbname,
    OUTPUT SEGMENT=SEGD.
```
Processing considerations for randomization of root segments to unprocessed areas

When the UNLOAD command is executed under normal conditions, processing continues if any root segments are randomized to an area in the DEDB that is not being processed by the command set and therefore bypassed and uncopied.

The optional BYPASS_RECORD keyword can be used to terminate the unload function when root segments are randomized to an area in the DEDB that is not being processed by the command set.

The following values are available for the BYPASS_RECORD keyword:

- **BYPASS_RECORD=NO** causes processing to terminate if any root segment is randomized to an area in the DEDB that is not being processed. An error message is generated that contains the key of the bypassed root segment.

- **BYPASS_RECORD=YES** (default) enables the unload function to continue processing when a root segment is randomized to an area in the DEDB that is not being processed. An informational message is generated by the unload function that includes a count of the total number of bypassed records.

In the example shown in **Figure 128**, any root segments that are randomized to an area other than *areanam1*, *areanam4*, or *areanam5* due to changes in the DBD or randomizer will cause the unload function to terminate.

**Figure 128** Using BYPASS_RECORD keyword to terminate UNLOAD processing when a root randomizes to an area not specified on OAREA keyword

```
UNLOAD DBD=dbdname,IAREA=(areanam1,areanam4,areanam5),
     OAREA=(areanam1,areanam4,areanam5),BYPASS_RECORD=NO
```
Samples of INCLUDE, EXCLUDE, and OUTPUT subcommands

Table 14 shows examples of the UNLOAD command combined with the INCLUDE, EXCLUDE, and OUTPUT subcommands and the resulting selected segments. These examples use the segment hierarchy shown in Figure 122.

<table>
<thead>
<tr>
<th>Command set</th>
<th>Selected segments</th>
<th>Output segments</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNLOAD IAREA=ALL</td>
<td>All roots and their dependent segments are selected.</td>
<td>A segment record is written for each selected segment.</td>
</tr>
<tr>
<td>no INCLUDE, EXCLUDE, or OUTPUT subcommands specified</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXCLUDE SEGMENT=SEGA</td>
<td>No segments are selected.</td>
<td>No segment records are output.</td>
</tr>
<tr>
<td>EXCLUDE SEGMENT= SEGA INCLUDE SEGMENT=SEGB</td>
<td>No segments are selected. (Excluding a parent segment excludes all of its dependent segments.)</td>
<td>No segment records are output.</td>
</tr>
<tr>
<td>EXCLUDE SEGMENT=SEGB</td>
<td>SEGA, SEGE, SEGF, and SEGG are selected.</td>
<td>A segment record is written for each selected segment.</td>
</tr>
<tr>
<td>INCLUDE SEGMENT=SEGA, WHERE=(selection criteria)</td>
<td>All SEG segments that meet criteria and their dependents are selected.</td>
<td>A segment record is written for each selected segment.</td>
</tr>
<tr>
<td>EXCLUDE SEGMENT=SEGB</td>
<td>SEGA and SEGG are selected.</td>
<td>All SEGA and SEGG segments selected are written.</td>
</tr>
<tr>
<td>EXCLUDE SEGMENT=SEGE</td>
<td>SEGA and SEGG are selected.</td>
<td>All SEGA and SEGG segments selected are written.</td>
</tr>
<tr>
<td>OUTPUT SEGMENT=SEGG</td>
<td>All roots and their dependent segments are selected.</td>
<td>All segments are written.</td>
</tr>
<tr>
<td>INCLUDE SEGMENT=SEGB, WHERE=(selection criteria)</td>
<td>All SEG, SEGE, SEGF, and SEGG segments are selected. Only SEGB (and SEGD children) segments that meet the INCLUDE selection criteria are selected.</td>
<td>All SEG, SEGE, SEGF, and SEGG segments are written. All SEG (and SEGD children) segments that meet the selection criteria are written.</td>
</tr>
</tbody>
</table>

Controlling error tolerance

A segment might contain an error that prevents the unload function from processing root or dependent segment chains. If such an error is encountered in a segment, the unload function terminates immediately.

The ERROR_THRESHOLD keyword can be used to enable the unload function to encounter and bypass a specified number of segments containing pointer errors without terminating.
When ERROR_THRESHOLD is used with the UNLOAD command, the database record is unloaded to the output file up to the point of the segment in error, and processing continues with the next accessible segment. The segment containing the pointer error (and its dependents) is bypassed, and is not written to the unload file. When the number of errors encountered in all database records that have been read exceeds the value specified on the ERROR_THRESHOLD keyword, processing of the area terminates. If an unload function terminates because of pointer errors, requested image copies will still be created.

When the default value of 0 (zero) is in effect for the ERROR_THRESHOLD keyword, the unload function terminates when the first pointer error is encountered.

The example in Figure 129 requests a DEDB unload and will bypass up to three segments containing pointer errors before terminating the unload function. The three segments containing pointer errors (and their dependent segments) are not written out, and unload processing continues with the next accessible segment. If a fourth pointer error is encountered the unload function terminates.

**Figure 129 Using the ERROR_THRESHOLD keyword to control error tolerance**

```
UNLOAD DBD=dbdname,ERROR_THRESHOLD=3
```

When you specify SDEP_PROCESS=PHYSICAL on the UNLOAD command, the process does not read SDEPs by following pointers. If you want the ERROR_THRESHOLD keyword to detect and bypass any pointer errors detected in the SDEP portion of the DEDB, then you must specify SDEP_PROCESS=LOGICAL.

**WARNING**

If your application relies on the marker segment concept, specifying SDEP_PROCESS=LOGICAL in conjunction with ERROR_THRESHOLD might result in the dropping or relocation of marker segments. For more information about marker segments, see page 186.

Making an image copy during an unload

You can request that one or more image copy data sets be produced during the unload function. Specify the IC subcommand for each image copy to be produced. For details about the IC subcommand, see Chapter 10, “Making an image copy.”

The example in Figure 130 requests the creation of a single output image copy data set for each area being unloaded.
Enhancing performance

The following methods might significantly improve the performance of the unload function:

- input caching
- input thread processing

Unload function input caching

The effect of input caching on the unload function depends on the following factors:

- type of input (image copy or area data set)
- type of caching that is used
- how SDEP segments (if any) are processed

Input caching with the ICACHE keyword during the unload function might significantly speed processing. A dataspace is used to implement the cache. The function has the same considerations as input caching for the change function. For information on these considerations, see “Enhancing performance” on page 163.

Unload function input thread processing

Fast Path Offline Suite products include thread processing features that let you input and output DEDB areas in parallel for parallel processing. Processing multiple areas of a DEDB simultaneously can improve the performance. The degree of performance enhancement that is provided by these features is dependent on the characteristics of your DEDB and the resources of your site.
The INPUT_THREADS keyword lets you specify the parallel input of a DEDB to the unload function. The INPUT_THREADS keyword lets you specify the maximum number of input areas to process concurrently. If resources are not available to support the specified number, parallel processing occurs only on the number that resources can support.

The following considerations apply to using INPUT_THREADS as a means of enhancing the unload function:

- The default (maximum) value for INPUT_THREADS is determined automatically by the product, based on the number of input areas, CPU processors, and other system resources.

- Specifying a number less than the automatic default can improve performance, but specifying a number greater than the default value will be ignored.

Ensure that adequate resources are available to support your selection.

## Reload function inputs and outputs

As shown in Figure 131, the reload function processes input from an input unload file. Use the reload function to load one or more areas of a DEDB. The ACB of the database being reloaded is used to describe the output areas and is referenced by the IMSACB DD statement.

**Figure 131  Reload function inputs and outputs**

[Diagram showing the process of reload function inputs and outputs]
The input to the reload function is a sequential input file for each area to be reloaded. The reload function accepts any of the supported unload formats as input, and will automatically determine which format it is reading.

The reload function accepts different formats (separate area DDs) in the same job. However, multiple areas of differing formats cannot be concatenated on a single IAREAxxx or IARxxxxx DD statement. When multiple input formats are used, each input area requires its own IAREAxxx DD or IARxxxxx DD statement.

The output from the reload function is the area data sets that are referenced by the areaname DD statements in the JCL. Any (or all) areas of the selected DEDB can be reloaded.

An optional output file (discard file) that is used to hold data that cannot be reloaded because of segment exceptions can also be created. For more information, see “Handling exceptions” on page 225.

The Input Area Summary Report, Output Area Summary Report, and the Database Summary Report are generated by the reload function to summarize reload activity. For an example of these reports, see the PFPRELD member in the REPORTS data set.

## Reload DBRC considerations

When DBRC is active during execution of the reload function and the output area is registered with DBRC, Fast Path Reorg/EP verifies from DBRC that each output area is marked “Recovery Needed.” These conditions are shown in Table 15. The reload function obtains exclusive access (EX) from DBRC for each output data set. No DBRC authorization is performed on the input data sets.

<table>
<thead>
<tr>
<th>Command</th>
<th>Authorization</th>
<th>Recovery Needed flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELOAD (area data set input)</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>RELOAD (outputs)</td>
<td>exclusive access (EX)</td>
<td>ON</td>
</tr>
</tbody>
</table>

If your areas do not reflect the appropriate “Recovery Needed” condition, their status must be changed before execution of the reload function. To set the “Recovery Needed” flag ON, execute the DBRC utility by using the command shown in Figure 132.

**Figure 132  DBRC command for setting the Recovery Needed flag**

```
CHANGE DBDS DBD(dbdname) AREA (areanamel) RECOV
```
Multiple area data sets (MADS) are not supported. The reload function reloads only the first area data set that is registered in the ADS list (in collating sequence by ddname). All other area data sets are marked unavailable. When reload processing has completed, use the IBM Online MADS Create utility to resynchronize the other (unavailable) area data sets.

When reload processing is completed, the reload function resets the “Recovery Needed” flag and notifies DBRC that an image copy is required for each area by setting the “Image Copy Needed” flag. You can use the IC subcommand to request that an image copy be created for each area during reload processing. DBRC is informed (NOTIFY.IC) when this image copy is created successfully, which resets the “Image Copy Needed” flag.

If you do not take an image copy during the reload process, an informational message is issued for each output area to indicate that an image copy should be taken before each area can be updated online.

**NOTE**

The reload process also creates a REORG record in DBRC, if DBRC detects that this record is required for recovery.

## RELOAD command keywords and subcommands

Table 16 lists the keywords and subcommands that are available for the RELOAD command.

<table>
<thead>
<tr>
<th>Function</th>
<th>Command or subcommand</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>selecting the database and areas</td>
<td>RELOAD</td>
<td>DBD IAREA OAREA</td>
</tr>
<tr>
<td>allocating the input area data sets</td>
<td>RELOAD</td>
<td>INPUT_DSN_MASK</td>
</tr>
<tr>
<td>allocating the output area data sets</td>
<td>RELOAD</td>
<td>OUTPUT_DSN_MASK</td>
</tr>
<tr>
<td>defining the output area data sets</td>
<td>ALLOCATE</td>
<td>all associated keywords</td>
</tr>
<tr>
<td>controlling free space in IOVF</td>
<td>RELOAD</td>
<td>IOVF_LOAD_HWM</td>
</tr>
</tbody>
</table>
Table 16  RELOAD command keywords and subcommands (part 2 of 2)

<table>
<thead>
<tr>
<th>Function</th>
<th>Command or subcommand</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>placing segments with load control</td>
<td>LOADCTL</td>
<td>all associated keywords</td>
</tr>
<tr>
<td>controlling segment expansion</td>
<td>RELOAD</td>
<td>EXPAND</td>
</tr>
<tr>
<td>controlling segment compression</td>
<td>RELOAD</td>
<td>COMPRESS</td>
</tr>
<tr>
<td>selecting SDEP processing method</td>
<td>RELOAD</td>
<td>SDEP_PROCESS</td>
</tr>
<tr>
<td>selecting database segments for input</td>
<td>INCLUDE EXCLUDE</td>
<td>SAMPLE_INTERVAL, SAMPLE_LIMIT, SEGMENT, WHERE</td>
</tr>
<tr>
<td>changing segment contents</td>
<td>OUTPUT</td>
<td>all associated keywords</td>
</tr>
<tr>
<td>controlling processing of root segments to non-processed areas</td>
<td>RELOAD</td>
<td>BYPASS_RECORD</td>
</tr>
<tr>
<td>handling exceptions</td>
<td>RELOAD</td>
<td>EXCEPTION_LIMIT keyword</td>
</tr>
<tr>
<td>handling discarded data</td>
<td>DISCARD_FILECTL</td>
<td>all associated keywords</td>
</tr>
<tr>
<td>rebuilding index(es) while reloading</td>
<td>IX</td>
<td>all associated keywords</td>
</tr>
<tr>
<td>analyzing the DEDB while reloading</td>
<td>RELOAD</td>
<td>POINTER_VALIDATION, RAP_VALIDATION, SDEP_VALIDATION, LARGEST_DATABASE_RECORDS</td>
</tr>
<tr>
<td></td>
<td>REPORT THRESHOLD</td>
<td>all associated keywords</td>
</tr>
<tr>
<td>making an image copy while reloading</td>
<td>IC</td>
<td>all associated keywords</td>
</tr>
<tr>
<td>enhancing performance</td>
<td>RELOAD</td>
<td>INPUT_THREADS, INDEX_THREADS, OCACHE, SORT, SORT_OPTION</td>
</tr>
</tbody>
</table>

For more information about the following topics, see the *Fast Path/EP Series Reference Manual*:

- syntax of commands, subcommands, and keywords that are discussed in this book
- diagrams that show the syntax and available parameters and values for Fast Path Offline Suite commands and subcommands
Specifying input and output

The DBD keyword identifies the name of the DEDB (DBD name) to be reloaded. The DBD keyword is required for reload processing.

Specifying areas to be read from unload input

The IAREA keyword identifies the names of the areas to be read from the input unload files. You can specify one or more area names. If you omit the IAREA keyword, all of the areas that are defined in the DEDB are read.

Multiple areas can be specified on the IAREA keyword by using any combination of area names, area numbers, or a area ranges. The following parameters are available for the IAREA keyword:

- **IAREA=ALL** (default) or **IAREA=*** specifies all areas of the DEDB.
- **IAREA=areaname** specifies one or more areas by using the one-character to eight-character area name for each area specified. Multiple area names must be enclosed in parentheses and separated by commas.
- **IAREA=areanumber** specifies one or more areas by using the one-character to five-character area number for each area specified. Multiple area numbers must be enclosed in parentheses and separated by commas.
- **IAREA=(RANGE=(startarea,endarea))** specifies a consecutive range of areas using either areaname or areanumber parameters. The area associated with startarea must be less than the area number associated with endarea.

An asterisk (*) can be used to specify all areas of the DEDB. When the * character is used with the RANGE keyword, it can be used to specify the beginning or ending range for specific areas of the DEDB.

To reload an entire DEDB, use a command set like the example shown in **Figure 133**.

**Figure 133 Sample control statement for reloading all areas from the input unload files**

```
RELOAD DBD=dbdname, IAREA=ALL, OAREA=ALL
```

To reload specific areas, use a command set like the example shown in **Figure 134**.
Specifying areas to be reloaded

The OAREA keyword identifies the names of the areas to be reloaded. You can specify one or more area names. If you omit the OAREA keyword, the RELOAD command will attempt to process all areas that are defined in the DBD.

Multiple areas can be specified on the OAREA keyword by using any combination of area names, area numbers, or a range of areas. The following parameters are available for the OAREA keyword:

- **OAREA=ALL** (default) or **OAREA=*** specifies all areas of the DEDB.

- **OAREA=areaname** specifies one or more areas by using the one-character to eight-character area name for each area specified. Multiple area names must be enclosed in parentheses and separated by commas.

- **OAREA=areanumber** specifies one or more areas by using the one-character to five-character area number for each area specified. Multiple area numbers must be enclosed in parentheses and separated by commas.

- **OAREA=(RANGE=(startarea,endarea))** specifies a consecutive range of areas using either areaname or areanumber parameters. The area associated with *startarea* must be less than the area number associated with *endarea*.

An asterisk (*) can be used to specify all areas of the DEDB. When the * character is used with the RANGE keyword, it can be used to specify the beginning or ending range for specific areas of the DEDB.

Values for the OAREA keyword function in the same manner as for the IAREA keyword. See the examples shown for the IAREA keyword in Figure 133 and Figure 134.
Allocating input area data sets

One input unload data set is required for each area name that is specified by using the IAREA keyword. Because the unload function (by default) creates one unload data set for each area in the database, the reload function (by default) expects one input data set for each area.

The input data sets can be supplied in the JCL or can be accessed using dynamic allocation. The DD statement can be named IAREAxxx, where xxx is the three-digit area number relating to the sequential order in which the areas are listed in the DBD, or IARxxxx, where xxxx is the five-digit area number between 00001 and 02048. If this DD statement is omitted from the JCL, the reload function tries to dynamically allocate it by using the INPUT_DSN_MASK keyword.

In some instances, there might not be a separate unload data set for each area in the database. For example, the OFILECTL subcommand might be used during the unload function to combine multiple areas into a single unload data set. When converting an HDAM database to a DEDB, the IBM HD Reorganization Unload utility creates a single unload data set, even though the target DEDB might contain multiple areas.

To handle such situations, the number of area names that you specify by using the IAREA keyword must match the number of input data sets. The area names that you use are arbitrary and affect only the ddname (IAREAxxx or IARxxxx) that used for the input files. The reload function does not confirm that input data which is read from an unload file corresponds to the area named. As each root segment is read from the input file, the randomizer is invoked to determine the output area into which the record is placed.

The reload function can read any input unload file format produced by the unload function, including HDUNLOAD, TFMT, MSDBINIT, and DBT. You do not need a keyword to specify the format of the input file. The reload function interrogates the file to determine the input unload file format and then reloads the data accordingly.

**NOTE**

An HDUNLOAD file format that is created by unloading an HALDB database can be used as input to the reload process, which allows an HALDB database to be converted to a DEDB. This file cannot be used as input into FABEUR6 and FABEUR7.

The example in Figure 135 shows a reload function by using two input data sets. Even though the database contains more than two areas, the data for all areas is contained within these two data sets. The DD statements that are used for the input data sets are named IAREA002 and IAREA005.
Allocating output area data sets

You can allocate the output area for the RELOAD command by using traditional DD statements or by using dynamic allocation. When using dynamic allocation, you can also specify certain processing criteria and allocation characteristics for the output area.

Using DD statements

You can use the following DD statements to allocate the output area for the RELOAD command.

**Areaname**

The areaname DD statement identifies the area data set to be created. If the areaname DD statement is omitted from the JCL, Fast Path Reorg/EP tries to dynamically allocate it.

The area name that is specified on the areaname DD statement must be the same as the area name that is specified in the ACB library. Fast Path Reorg/EP verifies this match by using the IMSACB DD statement.

**PFPTOTAL**

The PFPTOTAL DD statement allocates an optional output PFPTOTAL data set. This output data set can be used as an audit file to verify segment output totals after using the reload process.

The PFPTOTAL data set is designed for input into a user-written program. For more information about the PFPTOTAL DD statement and the record format, see the *Fast Path/EP Series Reference Manual*. 

---

**Figure 135  Reloading with two input data sets**

```
RELOAD DBD=dbname, OAREA=ALL,
       IAREA=(areaname2,areaname5)
```
Using dynamic allocation to name the output area data set

If you are using dynamic allocation to define the output area, do not include the areaname DD statement. Fast Path Reorg/EP tries to obtain the data set name for allocation in the following order:

- If the OUTPUT_DSN_Mask keyword is specified, it is used to generate the data set name.
- If DBRC is active and the area is registered, the registered area data set name is obtained from DBRC.
- The STEPLIB is searched for the DFSMDA member that contains the data set name for this area.

If DBRC is active and the area is registered with DBRC, the allocated data set name must match the registered data set name regardless of how it is allocated.

Using the ALLOCATE subcommand to customize dynamically defined output areas

If you elect to dynamically define the output areas for the RELOAD command, you can specify unique allocation characteristics for the output areas by using the ALLOCATE subcommand and its related keywords. The ALLOCATE subcommand can be used as a replacement for DELETE, DEFINE, and other statements that are normally specified by executing the Access Method Services IDCAMS program.

**NOTE**

If the IDCAMS program is already defined in your job input, it is not necessary to remove it. However, any defining statements that are specified under IDCAMS for the output area are superseded by similar parameters that are specified on the ALLOCATE subcommand.

By applying the functionality available with the ALLOCATE subcommand, you can accomplish the following tasks and performance improvements:

- automatically adjust the VSAM cluster definition to match an area’s size characteristics
- help to simplify JCL streams and job restarts by combining VSAM cluster definition (traditionally defined by IDCAMS) and area processing into a single step
- improve the area’s usage of fragmented DASD space and irregular sized extents
- enhance system I/O performance by promoting parallel processing against the area by easily separating the RAA, IOVF, and SDEP portions of an area across multiple volumes

The ALLOCATE subcommand lets you perform the following tasks:

- delete and define, reuse, or rename the VSAM cluster that is used for the output areas
- pass a sequential or partitioned data set to Access Method Services to be used as input for allocation of the new VSAM cluster
- specify the number of volumes and allocation units for the output areas
- specify the number of volumes and allocation units for the RAA, IOVF, and SDEP portions of the output areas
- pass optional parameters to Access Method Services for the VSAM cluster definition
- specify the SMS storage class, management class, and data class for the output areas
- specify space requirements for the output areas
- specify volume serial identifiers for the output areas

These functions are defined by specifying combinations of keywords that are specified on the ALLOCATE subcommand.

**Identifying the areas for ALLOCATE processing**

The OAREA keyword is specified on the ALLOCATE subcommand to identify the areas for which you want to customize allocation. You can specify one or more areas on a single ALLOCATE subcommand, or specify separate ALLOCATE subcommands for any (or all) of the output areas that are identified on the OAREA keyword under the RELOAD command.

Areas can be specified on the OAREA keyword under the ALLOCATE subcommand by using any combination of area names, area numbers, or a area ranges. For a description of valid OAREA keyword parameters, see “Selecting areas for output” on page 178.
Specifying the area to be used

The ACTUATE keyword requests optional VSAM cluster processing. The following parameters are available:

- ACTUATE=DELETE causes the current VSAM cluster to be deleted before the new VSAM cluster is allocated.
- ACTUATE=REUSE causes the current VSAM cluster to be allocated and overwritten without allocating a new VSAM cluster.
- ACTUATE=RENAME causes the current VSAM cluster name to be altered according to the data set name mask that you specify before allocating the new VSAM cluster.
- ACTUATE=IDCAMS causes a sequential or partitioned data set to be passed to Access Method Services as input for the allocation of the new VSAM cluster. Due to the design of this feature, each area should have a separate IDCAMS definition.

If you omit the ACTUATE keyword, it is assumed that the output area does not already exist. A new output area will be allocated.

You can also specify the EROPT parameter on the ACTUATE keyword to specify whether ALLOCATE subcommand processing should continue if a processing error is encountered. Specify EROPT=IGNORE to continue processing. Specify EROPT=ABORT to terminate processing.

Allocating the output area and specific portions of the area

Four keywords are available on the ALLOCATE subcommand that let you control the following allocation characteristics for the specified output area or portion of the output area:

- size of each extent for the area or portion of the area
- number of extents (volumes) that the area or portion of the area will span

By default, the entire output area or any portions of an output area (RAA, IOVF or SDEP) that are not explicitly specified with extent size or volume count parameters will be contained on one volume.

The size of the extents are controlled by specifying one or more values on the following keywords:

- The CONFIGURE_AREA keyword specifies the size of the extent or extents for the entire area data set.
The CONFIGURE_RAA keyword specifies the size of the extent or extents for the root addressable portion of the area data set.

The CONFIGURE_IOVF keyword specifies the size of the extent or extents for the independent overflow portion of the area data set.

The CONFIGURE_SDEP keyword specifies the size of the extent or extents for the sequential dependent portion of the area data set.

The values that are specified on the CONFIGURE_AREA keyword or any combination of the CONFIGURE_RAA, CONFIGURE_IOVF and CONFIGURE_SDEP keywords represent the amount of track or cylinder space to be allocated to the extent. Selection of tracks or cylinders for the extents is requested by specifying the SPACE keyword parameter on the ALLOCATE subcommand or is determined by IDCAMS.

To define multiple extents with the same amount of space, you can specify a value on the VOLCNT parameter with an extent size value. The value that is specified on the VOLCNT parameter indicates the number of extents (volumes) that the area or portion of the area will span. Each extent is sized according to the specified extent size.

You can specify the IDCAMS_OPTION keyword on the ALLOCATE subcommand to pass optional parameters to Access Method Services for the VSAM cluster definition. Numerous parameters are available for use with this keyword to control the values that are used by AMS for the VSAM cluster definition. For more information, see the IBM Access Method Services Reference Manual.

**Specifying space requirements, class, and volume identifier for the output area**

Other keywords let you specify the SMS class, space requirements, and volume serial identifier for the output area:

- AVGREC
- DATACLAS
- MGMTCLAS
- SPACE
- STORCLAS
- VOLSER

These keywords function in the same manner as on other Fast Path/EP subcommands.
ALLOCATE subcommand sample scenarios

For scenarios that show how the ALLOCATE subcommand can be used with key related keywords to define allocation parameters for areas that are output by RELOAD, CHANGE, and INITIALIZE commands, see Appendix C, “Sample command scenarios.”

Reloading an area from multiple input files

When using the reload function to load an area of a DEDB, sometimes you need to reload the area by using multiple input files. These input files might be created for various reasons. For example, data could be unloaded from several existing databases and used as input to create a new database.

The following restrictions apply to the RELOAD command process when reloading an area from multiple input files:

- If duplicate root segments are encountered, the reload will terminate. However, the reload will not terminate if the EXCEPTION_LIMIT keyword is specified with a value that is greater than the number of segment errors encountered. For more information about using the EXCEPTION_LIMIT keyword, see “Handling exceptions” on page 225 and the Fast Path/EP Series Reference Manual.

- If root segments have been rerandomized, and there are SDEP segments in the area, then SDEP_PROCESS=LOGICAL must be specified. SDEP_PROCESS=PHYSICAL is not allowed when root segments are rerandomized to another area.

To reload an area from multiple input files, use one of the following methods:

- Concatenate the input files on the IAREAxxx or IARxxxx DD statements for the area when executing the reload. Before proceeding with this method, ensure that the input file format is the same for each of the unload files.

- Execute separate reloads for each input file. The reloads must be run consecutively, rather than concurrently, because of the JCL disposition of DISP=OLD for the area data set. If you use this method, do not perform a VSAM cluster delete/define between the reload steps or jobs.
Controlling free space in IOVF

When IOVF space is needed to accommodate a RELOAD process, the normal space search algorithm places segments sequentially into IOVF blocks. The placement of segments within IOVF in this manner can affect application performance by causing longer reads of IOVF to find available free space. It is sometimes useful to influence the space search algorithm during the reload function to control the amount of space that is used in each IOVF block.

You can request a percentage of total space to be used in each IOVF block when the RELOAD command reloads segments for each root. The IOVF_LOAD_HWM keyword can be specified to request a percentage value between 50 and 100 (the default) of space to be filled in each IOVF block. During RELOAD command processing, Fast Path Reorg/EP will fill each IOVF block to the specified percentage before attempting to place segments in new storage locations.

If you request a value (percentage) less than 100 and the IOVF is not large enough to store all data at that fill level, Fast Path Reorg/EP will make a second load pass to load data until all data is stored or all IOVF blocks in the area are full.

The example in Figure 136 requests that each block of IOVF will be filled to 70 percent of its space capacity during reloading of the area. If the 70 percent of each IOVF block is not adequate for the amount of IOVF needed by the RELOAD, Fast Path Reorg/EP makes a second pass through IOVF and fill each sequential IOVF to 100 percent of capacity until the RELOAD command processing is completed. If all existing space in IOVF is used on the second pass, message BMC111167C is issued to indicate the out-of-space condition.

Placing segments with load control

The normal space search algorithm places segments into the control intervals in a tightly packed manner. The root segment is placed within the RAA block (if room is available) that is determined by the randomizer. Direct dependent segments belonging to the root are placed into the same RAA block or DOVF blocks within the UOW until they are all used. IOVF blocks are selected and filled as needed.
The placement of segments within the types of control intervals (RAA, DOVF, and IOVF) has a significant effect on performance. It is sometimes useful to influence the space search algorithm during the reload function to relegate infrequently used segments into IOVF (or DOVF) blocks. Segment placement will preserve space in the RAA (or DOVF) blocks for more active data.

Placement of selected segments during the reload function can be accomplished by using the LOADCTL subcommand and its related keywords. The LOADCTL subcommand lets you perform the following tasks:

- select a segment, a segment and its dependents, or its dependents only for placement
- control the placement of selected segments into IOVF or DOVF
- control placement of segments, based on their data content
- control placement of segments, based on a specified number of segment occurrences

**NOTE**
To maintain this data placement, you must continue to use the same LOADCTL subcommand (or subcommands) for any subsequent reload processing on the DEDB.

Figure 137 shows a typical DEDB segment hierarchy. This hierarchy is used in subsequent examples in this section that show how the LOADCTL subcommand can be used with its associated keywords.

**Figure 137 Database hierarchical structure in LOADCTL examples**
Defining segments for load control placement

The SEGMENT keyword is required with the LOADCTL subcommand to identify segments for load control placement. The following subparameters are available for the SEGMENT keyword to provide selection versatility:

- SEGMENT=(segname,ONLY) (the default) specifies that the LOADCTL subcommand applies to the named segment only. Its dependent segments (if any) are not affected.

- SEGMENT=(segname,DEPENDENTS) specifies that the LOADCTL subcommand applies to the dependents of the named segment only. The named segment is not affected.

- SEGMENT=(segname,BOTH) specifies that the LOADCTL subcommand applies to both the named segment and to all of its dependents.

The LOCATION keyword specifies where you want the segments placed:

- LOCATION=IOVF (the default) places segments in IOVF storage, regardless of the amount of space that is available in RAA and/or DOVF blocks.

- LOCATION=DOVF places segments in DOVF storage regardless of the amount of space that is available in the RAA block. If there is not enough space in the DOVF blocks, the segment is placed into IOVF.

Placing a specific segment type

If you specify the SEGMENT keyword with the ONLY subparameter (the default), the named segment only is placed by using load control.

In Figure 138, all occurrences of SEGB are placed into IOVF.

Figure 138  LOADCTL with SEGMENT only

```
RELOAD DBD=dbdname
    LOADCTL SEGMENT=(SEGB,ONLY),LOCATION=IOVF
```

Placing dependents of a segment type

If you specify the SEGMENT keyword with the DEPENDENTS subparameter, all dependents of the named segment are placed by using load control. The named segment is placed normally.
In the example in Figure 139, the DEPENDENTS subparameter is specified. Segment placement control is applied to all dependent segment types of the named segment (SEGB). All dependent segment occurrences (SEGC, SEGD, SEGE, and SEGF) are placed within an IOVF block, regardless of the amount of space that might remain within the RAA and DOVF blocks.

**Figure 139  Sample LOADCTL for segment dependents**

```plaintext
RELOAD DBD=dbname,AREA=areaname
    LOADCTL SEGMENT=(SEGB,DEPENDENTS)
```

**Placing a segment and its dependents**

If you specify the SEGMENT keyword with the BOTH subparameter, the named segment and all of its dependents are placed by using load control.

In the example in Figure 140, the BOTH subparameter is specified. Segment placement control is applied to SEGG and its dependent segment (SEGH).

**Figure 140  Sample LOADCTL for segment and its dependents**

```plaintext
RELOAD DBD=dbname,AREA=areaname
    LOADCTL SEGMENT=(SEGG,BOTH),LOCATION=DOVF
```

**Placement control using segment content**

You can specify that placement control is to be applied, based on the data values that are contained in segments, by specifying the WHERE keyword. The WHERE keyword lets you specify field names or field positions and lengths, an operand, a value, and optional Boolean operators. For information about the operations and syntax of the WHERE keyword and syntax rules for coding these expressions, see the Fast Path/EP Series Reference Manual.

**Placing specific segments based on segment data**

You can specify that placement control is to be applied, depending on a value or values that appear within the named segment data. The example in Figure 141 requests that placement control be applied to the segment type that is specified explicitly by the SEGMENT keyword. Selection criteria is specified by using the WHERE keyword.

**Figure 141  Sample LOADCTL basic selection criteria**

```plaintext
RELOAD DBD=dbname
    LOADCTL SEGMENT=(SEGB,ONLY),LOCATION=IOVF,
    WHERE=(3:1 EQ 'X')
```
If the criteria is met, SEGB is placed within an IOVF block, regardless of the amount of space that might remain within the RAA and DOVF blocks. If the criteria is not met, the segment occurrence is placed normally.

### Placing specific segments based on segment data in a different segment

You can specify that criteria be based on data within a different segment with the WHERE keyword by using a qualified field reference. The qualified field must lie within the segment parentage path from the root segment to the segment that is specified by using the SEGMENT keyword.

The example in Figure 142 requests that placement control be applied to SEGD specified by the SEGMENT keyword. Selection criteria is specified by using the WHERE keyword that references a field within the parent segment (SEGB). If the criteria is met, the SEGD segment occurrences are placed within an IOVF block, regardless of the amount of space that might remain within the RAA and DOVF blocks. If the criteria is not met, the SEGD segments are placed normally.

![Figure 142 Sample LOADCTL using qualified selection criteria](image)

```sql
RELOAD DBD=dbdname, IAREA=areaname
LOADCTL SEGMENT=SEGD,LOCATION=IOVF,
   WHERE=(SEGB.3:1 EQ 'X')
```

### Placing multiple segments based on different criteria

You can specify as many LOADCTL subcommands as are required to obtain the desired segment placement.

Figure 143 requests that placement control be used for four segments. The WHERE keyword is used for each segment to specify a Boolean expression that controls the placement of each segment occurrence.

![Figure 143 Using multiple LOADCTL subcommands to control segment placement](image)

```sql
RELOAD DBD=dbdname, IAREA=areaname
LOADCTL SEGMENT=SEGB,LOCATION=IOVF,
   WHERE=(3:2 EQ C'80'
       OR 3:2 EQ C'90')
LOADCTL SEGMENT=SEGF,LOCATION=IOVF,
   WHERE=(SEGA.16:1 EQ C'N')
LOADCTL SEGMENT=SEGG,LOCATION=IOVF,
   WHERE=(16:4P LE P'19970101' AND
       20:1 EQ C'Y')
LOADCTL SEGMENT=SEGI,LOCATION=IOVF,
   WHERE=(13:2X GT X'4040')
```
If multiple LOADCTL subcommands are specified for the same segment type, they are evaluated in the order in which they are specified. The first LOADCTL subcommand for the segment for which the WHERE criteria is met (or if no WHERE criteria is specified) are used to control placement.

Figure 144 shows an example of multiple LOADCTL subcommands that are specified for the same segment. If the first WHERE criteria is met, SEGB is placed in IOVF. If the second WHERE criteria is met, SEGB is placed in DOVF. If neither WHERE criteria is met, the segment is placed normally.

**Figure 144  Sample LOADCTL using multiple WHERE statements for the same segment type**

```
RELOAD DBD=dbdname, IAREA=areaname
LOADCTL SEGMENT=SEGB, WHERE=(3:2 EQ '80'), LOCATION=IOVF
LOADCTL SEGMENT=SEGB, WHERE=(3:2 GE '80'), LOCATION=DOVF
```

**Placement control using segment counts**

You can specify that placement control be applied after a number of segment occurrences for a root segment have been processed by using the INSERT_LIMIT_COUNT keyword.

The example in Figure 145 requests that placement control be used for a specified segment type. The first five segments of this type for each root segment are placed normally. The sixth and all subsequent occurrences of the segment type are placed within an IOVF block, regardless of the amount of space that might remain within the RAA and DOVF blocks within the UOW.

**Figure 145  Sample LOADCTL using segment occurrences**

```
RELOAD DBD=dbdname, IAREA=areaname
LOADCTL SEGMENT=SEGE, LOCATION=IOVF, INSERT_LIMIT_COUNT=5
```

The example in Figure 146 combines the WHERE and INSERT_LIMIT_COUNT keywords on a single LOADCTL subcommand. All segment types that do not meet the WHERE criteria are placed normally. Of the segments that do meet the WHERE criteria, the first three are placed normally. The fourth and all subsequent occurrences of the segment type that meet the WHERE criteria are placed within an IOVF block, regardless of the amount of space that might remain within the RAA and DOVF blocks within the UOW.
Controlling segment expansion

As an option, you can expand compressed segment data from the unload file before it is compressed and written to the area. This option is determined by the EXPAND keyword.

- **EXPAND=NO** (the default) causes segment data to be written to the output area in the same format as it appears in the unload file.

- **EXPAND=YES** causes compressed segment data to be expanded before it is recompressed and written to the output area.

- **EXPAND=(segment1, segment2, ..., segmentn)** causes only the compressed segments which are listed by name to be expanded before they are recompressed and written to the output area.

Figure 147 shows an example of the EXPAND keyword with the RELOAD command.

**Figure 147 Using the EXPAND keyword with reload**

```plaintext
RELOAD DBD=dbname, IAREA=areaname
LOADCTL SEGMENT=SEGE, LOCATION=IOVF,
WHERE=(3:2 EQ C'80'),
INSERT_LIMIT_COUNT=3
```

Figure 148 shows an example of using the EXPAND keyword to selectively specify segment expansion. Only **segment1** and **segment3** in **area1** are expanded and recompressed; all other segments are written to the output area in the same format as they appear in the unload file.

**Figure 148 Selectively specifying segments for expansion with reload**

```plaintext
RELOAD DBD=dbname, IAREA=area1,
OAREA=ALL, EXPAND=(segment1, segment3)
```
Controlling segment compression

As an option, you can compress segment data from the unload file before it is written to the output area during the reload function. This option is determined by the COMPRESS keyword.

- COMPRESS=NO (the default) causes segment data to be written to the output areas in the same form as it appears in the input unload file.

- COMPRESS=YES causes expanded segment data from the input unload file that is marked as expanded to be compressed before it is written to the output areas.

- COMPRESS=(segment1, segment2, ..., segmentn) causes only the expanded segments which are listed by name to be compressed before they are written to the output areas.

In the example shown in Figure 149, IAREA and OAREA default to ALL. The segments that have a compression exit that is specified in the DBD are compressed.

Figure 149  Using the COMPRESS keyword with reload

RELOAD DBD=dbdname,COMPRESS=YES

Figure 150 shows an example of using the COMPRESS keyword to selectively specify segment compression for specified area. Only segment1, segment3 and segment5 in area1 are compressed; all other segments are written to the output area in the same format as in the unload file.

Figure 150  Selectively specifying segments for compression with reload

RELOAD DBD=dbdname, IAREA=area1, OAREA=ALL, COMPRESS=(segment1,segment4,segment5)

Selecting SDEP processing method

If the unload file has an SDEP segment defined, you can optionally indicate how these segments are to be processed. Two methods are available for processing SDEP segments during execution of the reload function. The option used is determined by the SDEP_PROCESS keyword:

- SDEP_PROCESS=V5COMP causes SDEP segments to be processed with the root segment that owns them. The SDEP segments in each output area are physically reordered into the same sequence as the parent root segments without a timestamp. They retain the same logical order (entry sequence) within each database record.
Selecting database segments for input

The segments that are selected during input processing are forwarded to the output processing for segment formatting. By default, all root segments and their dependent segments are selected. You can modify the selection of segments by using the optional INCLUDE and EXCLUDE subcommands.

You can also use several keywords with the INCLUDE and EXCLUDE subcommands to further refine the selection process.

Including and excluding segments for input

The INCLUDE subcommand lets you specify criteria for selection of segments that will be included in the reloaded area. If the INCLUDE subcommand selection criteria are met, the selected segments and their dependents are written to the reloaded area.

The EXCLUDE subcommand lets you specify criteria for selection of segments that will be omitted from the reloaded area. If the EXCLUDE subcommand selection criteria are met, the selected segments and their dependents are not written to the reloaded area.

NOTE
When the SDEP_PROCESS=V5COMP value is specified on the RELOAD command, the unload file must have been created with SDEP_PROCESS=LOGICAL.

- SDEP_PROCESS=NONE causes SDEP segments that are present in the unload file to be omitted (not written to the output area). If SDEP segment types are detected in the unload file, the segments are not written to the reloaded area. An informational message is generated to indicate that SDEP segments will be removed.

Figure 151 shows an example of the SDEP_PROCESS keyword with the RELOAD command.
The SEGMENT keyword specifies the name of the segment to which the selection criteria are applied. Additional selection criteria can be specified by using the WHERE keyword, which uses field names or positional identifiers with relational or Boolean logical expressions.

A simple INCLUDE or EXCLUDE subcommand uses the required SEGMENT keyword without additional keywords. Understanding the following rules that apply to basic INCLUDE and EXCLUDE subcommands is prerequisite to using additional keywords with INCLUDE and EXCLUDE:

- By default, if no INCLUDE or EXCLUDE statement is used with the RELOAD command, all roots and dependents are selected for processing.

- The INCLUDE subcommand selects the segment that is identified by the SEGMENT keyword and all dependents of the segment. Including a segment in this manner also selects the parent hierarchy chain above the segment.

- The EXCLUDE subcommand excludes the segment that is identified by the SEGMENT keyword and all dependents of the segment. You cannot include a segment and exclude a root or a parent of the segment. Excluding a parent excludes the entire hierarchy beneath the parent.

A basic RELOAD command with an INCLUDE and EXCLUDE subcommand is shown in Figure 152.

### Figure 152  Using the RELOAD command with INCLUDE and EXCLUDE subcommands

```plaintext
RELOAD DBD=dbname, IAREA=(areanam1, areanam3), OAREA=(areanam1, areanam3)
  INCLUDE SEGMENT=segname1
  EXCLUDE SEGMENT=segname2
```

**Narrowing the selection**

While identifying the segments of interest for the reload using the INCLUDE and EXCLUDE subcommands, you can apply additional criteria to narrow the selection by using the following keywords:

- The WHERE keyword selects segments by applying conditional criteria that is specified in a Boolean expression.

---

**NOTE**

You cannot include a dependent segment and exclude its parent segment. Specifying a parent segment on the EXCLUDE subcommand also excludes all dependent segments.
The SAMPLE_INTERVAL keyword selects segments by intervals of occurrence.

The SAMPLE_LIMIT keyword sets a limit on the number of segments meeting the criteria that will be output to the reloaded area.

**Conditional criteria**

You can narrow segment selection by using the WHERE keyword. This combination applies relational and Boolean operators to apply conditional criteria when selecting segments. For information about the operations and syntax of the WHERE keyword and syntax rules for coding these expressions, see the *Fast Path/EP Series Reference Manual*.

Figure 153 shows a typical DEDB segment hierarchy. This hierarchy is used in subsequent examples in this section that show how the INCLUDE and EXCLUDE subcommands can be used with their associated keywords.

**Figure 153  SEGA root segment and its dependents**

The example in Figure 154 is used to create a test system database from a production version by selecting a specific set of records. The example selects only the SEGA segments that have the field ROOTKEY equal to BMCPFP. For the selected SEGA segments, only SEGB (and its dependents) where position 23 is equal to 22 will be selected. Only SEGE segments (of the selected SEGA segments) that were created after the year 1997 are selected. Although SEGG is defined in the DBD, it is excluded from the reloaded area.

**Figure 154  RELOAD command using INCLUDE, EXCLUDE, and WHERE**

```sql
RELOAD DBD=dbdname, IAREA=(areanam1,areanam3),
       OAREA=(areanam1)
       INCLUDE SEGMENT=SEGA,
       WHERE=(ROOTKEY EQ C'BMCPFP')
       INCLUDE SEGMENT=SEGB,
       WHERE=(23:2 EQ C'22')
       INCLUDE SEGMENT=SEGE,
       WHERE=(DATE GT C'1997')
       EXCLUDE SEGMENT=SEGG
```
Criteria that are applied to alternate segments

You can specify segment selection, based on criteria that are applied to alternate segments, by using a segment-qualified field reference within the WHERE keyword. Only the parentage segments within the hierarchical path from the root can be used for qualification.

In the example in Figure 155, SEGC is selected only if FIELD1 in SEGB is equal to the character string BMC.

Segment sampling

The SAMPLE_INTERVAL keyword can be used to specify segment selection based on the frequency of the segment occurrence. For example, if you have specified SAMPLE_INTERVAL=5, every fifth segment will be selected. For a root segment, every fifth occurrence in the database and all of its dependents will be selected. For a dependent segment, every fifth occurrence under its parent is selected.

The SAMPLE_LIMIT keyword can be used to specify the maximum number of segment occurrences to be selected. For example, if you have specified SAMPLE_LIMIT=500, the first 500 segments will be selected. For a root segment, the first 500 occurrences and all its dependents are selected. For a dependent segment, the first 500 occurrences within its parent are selected.

In the example in Figure 156, all areas are included in the process since the IAREA and OAREA keywords are not specified. The result of the INCLUDE subcommand is that every ninth occurrence of SEGA (and all its dependents) are written to the reloaded area, with a maximum of 5000 occurrences of SEGA. The EXCLUDE subcommand changes the result of the previous INCLUDE for dependent segment SEGE. For each selected SEGA segment, the first five segment occurrences for segment SEGE are omitted. If, for example, segment SEGE has eight occurrences, the last three segment occurrences would be written to the reloaded area. If segment SEGE had only four occurrences under another selected SEGA, no SEGE segments (or their dependents) would be written to the reloaded area.
The OUTPUT subcommand and its associated keywords can be used to change the content of the segment. By default, the output areas will contain the full, unmodified contents of the input segments.

To modify the segment contents, you can use the OUTPUT subcommand with the FIELDS keyword. The FIELDS keyword specifies a list of expressions; the value of each expression is evaluated and placed into the output segment in the order that the expressions are specified. The length of the segment is computed by the product as the sum of the lengths of the values produced from these expressions. For information about the operations and syntax of the FIELDS keyword and syntax rules for coding expressions, see the Fast Path/EP Series Reference Manual.

**WARNING**
When the FIELDS keyword is used with the OUTPUT subcommand to change the key of a dependent segment with a RELOAD process, the segments must remain in ascending key sequence under their parent. Fast Path Reorg/EP does not have a facility to resequence the dependent segments. The reload process terminates if the dependent segments are not maintained in ascending key sequence.

**NOTE**
When the FIELDS keyword is used with the OUTPUT subcommand to modify the content of an unloaded compressed segment with a RELOAD process, you must expand the segment by using the EXPAND=YES value or the EXPAND=(segment1,segment2, ...segmentn) syntax on the UNLOAD command. This expansion will ensure that the operands which are specified on the FIELDS keyword correspond to the proper segment columns.

To recompress the data before it is written to the output area, specify COMPRESS=YES or COMPRESS=(segment1,segment2, ...segmentn) on the RELOAD command. The format for the COMPRESS keyword should match the format of the EXPAND keyword.

Figure 157 shows an example where a literal value is being inserted into the output segment during a DEDB reload.
Processing considerations for randomization of root segments to unprocessed areas

The content of the output segment can be controlled by using conditional qualification that is specified by the optional WHERE keyword on the OUTPUT subcommand. If multiple OUTPUT subcommands are specified for the same segment, the first subcommand in which the selection criteria is satisfied is used. If the selection criteria is not satisfied for any OUTPUT subcommand for a segment, the input segment is copied to the output area without modification.

Figure 158 shows an example where the content of a segment is conditionally modified during a DEDB reload. If the WHERE condition that is specified on the OUTPUT subcommand is met, the FIELDS keyword modifies the segment content. If the condition is not met, the segment content is unchanged.

NOTE
Table 14 on page 195 provides a set of examples of various combinations of INCLUDE, EXCLUDE and OUTPUT subcommands that are specified for an unload process. These subcommands function the same way for the RELOAD command, and can be used as a reference for changing and selecting segments for a reload process.

Processing considerations for randomization of root segments to unprocessed areas

When the RELOAD command is executed under normal conditions, processing continues if any root segments are randomized to an area in the DEDB that is not being processed by the command set and therefore bypassed and uncopied.

The optional BYPASS_RECORD keyword can be used to terminate the reload function when root segments are randomized to an area in the DEDB that is not being processed by the command set.
The following values are available for the BYPASS_RECORD keyword:

- **BYPASS_RECORD=NO** causes processing to terminate if any root segment is randomized to an area in the DEDB that is not being processed. An error message is generated that contains the key of the bypassed root segment.

- **BYPASS_RECORD=YES** (default) enables the reload function to continue processing when a root segment is randomized to an area in the DEDB that is not being processed. An informational message is generated by the reload function that includes a count of the total number of bypassed records.

In the example shown in Figure 159, any root segments that are randomized to an area other than areanam1, areanam4, or areanam5 due to changes in the DBD or randomizer will cause the reload function to terminate.

**Figure 159  Using BYPASS_RECORD keyword to terminate RELOAD processing when a root randomizes to an area not specified on OAREA keyword**

```
RELOAD DBD=dbdname,IAREA=(areanam1,areanam4,areanam5),
OAREA=(areanam1,areanam4,areanam5),BYPASS_RECORD=NO
```

### Handling exceptions

By default, the first input data segment exception encountered causes the RELOAD command process to terminate. Fast Path Reorg/EP provides an exception handling facility that allows the RELOAD command to tolerate segment errors, allowing processing to continue. A *segment exception* or *segment error* is an input data segment that cannot be reloaded. A segment that cannot be reloaded, along with its dependent segments, can be bypassed and optionally written to a discard file. A *discard file* is an output file that is used to hold data that cannot be reloaded.

If an unload file contains data that cannot be reloaded, the exception handling facility for the RELOAD command can be used to tolerate this data and continue processing at the next valid data segment. When a segment exception is encountered, the reload process will bypass that segment and the dependent segments that follow until:

- another occurrence of the same segment type is encountered
- an occurrence of a segment at the same or higher level of the hierarchy is encountered
- end of the file is reached
To tolerate segment exceptions during the reload process and optionally write the discarded data to a discard file, you need to:

1. Assign the number of exceptions, to be allowed before the RELOAD command process terminates, to the EXCEPTION_LIMIT keyword. This assignment overrides the default value of zero, to allow reload processing to continue. Once the number of segment exceptions encountered is greater than the exception limit value assigned, reload processing terminates or stops. For more information, see “Assigning an exception limit” on page 226.

2. Specify the DISCARD_FILECTL subcommand, to optionally write the segment exceptions to a discard file. The EXCEPTION_LIMIT keyword and the DISCARD_FILECTL subcommand must be provided for the segment exceptions to be written to the discard file. For more information, see “Writing to a discarded data file” on page 227.

**NOTE**
When segment exceptions are encountered, the reload step will end with a return code of 12, whether or not segment exceptions have been discarded.

---

**Assigning an exception limit**

To override the RELOAD command default of no input segment errors, assign an exception limit value that is greater than zero to the EXCEPTION_LIMIT keyword. As long as the exception limit value assigned is greater than the number of segment exceptions encountered, the segment exceptions are bypassed and reloading of the area data set continues. For detailed information about the segment exceptions that could cause segments to be bypassed, see the discard file record layout in the Fast Path/EP Series Reference Manual.

The EXCEPTION_LIMIT keyword enables the reload process to tolerate a defined number of segment exceptions on an input area. This functionality allows the loading of the database to continue. The segments that are bypassed can be written to a discard file if the EXCEPTION_LIMIT keyword and the DISCARD_FILECTL subcommand are provided. If only the EXCEPTION_LIMIT keyword is provided, the segments are bypassed, but are not written to a discard file. For information about writing to a discard file, see “Writing to a discarded data file” on page 227.

The value that you assign to the EXCEPTION_LIMIT keyword represents the number of segment exceptions that will be tolerated for an input area. The default value, EXCEPTION_LIMIT=0, indicates that no segment exceptions will be tolerated, and the reload process will terminate when a segment exception is encountered. The EXCEPTION_LIMIT keyword can be assigned a numeric value up to 2,147,483,647. Reload processing will terminate when the number of segment exceptions
Writing to a discarded data file

The RELOAD command can write the segments that contain exceptions, along with their dependents, to an output discard file. This output file is used to hold data that cannot be reloaded.

The discard file name is specified by using the DISCARD_FILECTL subcommand. This discard file must be a valid file name, and can be predefined or dynamically allocated. The segments that are written to the discard file are in standard variable length record format. For more information about the discard file record layout, see the *Fast Path/EP Series Reference Manual*. 

NOTE

The value that is assigned to the EXCEPTION_LIMIT keyword applies to each input area.

When the number of segment exceptions encountered exceeds the value assigned to the EXCEPTION_LIMIT keyword, the following actions occur:

- An informational message is written to the PFPPRINT DD, indicating that the number of exceptions detected has exceeded the assigned value.

- RELOAD command processing terminates.

An example of assigning a value to the EXCEPTION_LIMIT keyword is shown in Figure 160. For more information about the EXCEPTION_LIMIT keyword, see the *Fast Path/EP Series Reference Manual*.

Figure 160  Sample syntax for assigning an exception limit value

```plaintext
RELOAD DBD=dbname,
   IAREA=area1, OAREA=area2,
   EXCEPTION_LIMIT=20
   DISCARD_FILECTL DSNAME='BMCPFP.reljob.DISCARD', DISP=(NEW, CATLG),
   SPACE=(CYL,1,1,RLSE), UNIT=DISK
```
The DISCARD_FILECTL subcommand and the EXCEPTION_LIMIT keyword must be provided to write segments that cannot be reloaded to the discard file. When only the EXCEPTION_LIMIT keyword is provided, the segments that cannot be reloaded are bypassed and are not written to a discard file. For more information about assigning an exception limit, see “Assigning an exception limit” on page 226 and the Fast Path/EP Series Reference Manual.

**NOTE**

Only one discard file is created for the input areas that are specified in the RELOAD command.

When segments are written to the discard file, an informational message is written to the PFPPRINT DD, indicating the total number of input area segments that have been discarded and written to the discard file.

**NOTE**

Segment exceptions that are excluded by data selection, such as INCLUDE or EXCLUDE, are not written to the discard file.

Examples of writing to a discard file by providing the DISCARD_FILECTL subcommand, along with the EXCEPTION_LIMIT keyword is shown in Figure 160 and in Figure 161. For more information about the DISCARD_FILECTL subcommand, see the Fast Path/EP Series Reference Manual.

Figure 161 Sample syntax for writing to a discard file

```plaintext
//DISCARD DD DISP=(,CATLG,DELETE) DSN=BMCPFP.reljob.DISCARD,
// UNIT=SYSDA,SPACE=(CYL,(10,5),RLSE)
//PFPSYSIN DD *
RELOAD DBD=bdname,
    EXCEPTION_LIMIT=INFINITE,
    IAREA=ALL,OAREA=ALL,
    INPUT_DSN_MASK='input.unload.file'
DISCARD_FILECTL DDNAME=DISCARD
```

**NOTE**

If you want to reload segment exceptions that have been written to the discard file, you will need to create a user-written program to reload the data. BMC does not supply a program to process the discard file.
Rebuilding indexes during reload

If the DEDB that is being changed has registered indexes that were created by using the BMC Fast Path Indexer/EP product, you can request that any (or all) of these indexes be rebuilt simultaneously with the reload function.

Specify the IX subcommand and associated keywords for the index to be rebuilt. For details about the IX subcommand, see the Fast Path Indexer/EP User Guide.

The example in Figure 162 requests that all indexes registered to the DEDB be rebuilt during the reload process.

**Figure 162  Requesting index rebuild during DEDB reload**

```
RELOAD DBD=dbname,IAREA=ALL,OAREA=ALL
 IX
```

The INDEX_THREADS keyword can be specified on the RELOAD command to specify the number of index threads to be used to rebuild indexes that are identified by an IX subcommand. The example in Figure 163 requests that five threads be used for rebuilding all indexes registered to the source database during the reload process.

**Figure 163  Requesting index threading during DEDB reload**

```
RELOAD DBD=dbname,IAREA=ALL,INDEX_THREADS=5
 IX
```

For more information about the INDEX_THREADS keyword, see the Fast Path Indexer/EP User Guide.

Analyzing the DEDB during reload

If your site has a license for the Fast Path Analyzer/EP product, pointer validation will occur automatically (by default) when you execute a database reload. This automatic process applies the default value of QUICK for the POINTER_VALIDATION keyword to perform a checksum validation of pointers for each segment type within each UOW of the database that is specified on the RELOAD command.

Automatic pointer validation provides assurance of the area’s pointer integrity, while providing statistics that show how the reload process affected space usage.
If an SDEP segment is defined for the database, SDEP pointers are also validated by default using the same technique as specified for direct pointers. You can manually specify another valid value for the SDEP_VALIDATION keyword to control how SDEP pointers are validated.

You can also specify additional functions that are associated with the analysis function:

- Use the RAP_VALIDATION keyword to control how RAPs are processed.
- Use the LARGEST_DATABASE_RECORDS keyword to specify the number of largest database records to be tracked by the analysis process.
- Use the REPORT and THRESHOLD subcommands to control the generation of analysis reports and exception testing.

The example in Figure 164 requests that analysis be performed on all output areas that are created by the RELOAD command. Direct pointers are validated by specifying FULL for the POINTER_VALIDATION keyword. SDEP pointers (if any exist) are validated by using the same technique. For more detailed information, see “Pointer validation” on page 97. Default reports are produced, and no threshold checking is performed.

Figure 164  Control Statements for Requesting Analysis during Reload

```
RELOAD DBD=dbdname,
   POINTER_VALIDATION=FULL
```

### Making an image copy during reload

You can request that one or more image copy data sets be produced during the reload function. Specify the IC subcommand for each image copy to be produced. For details about the IC subcommand, see Chapter 10, “Making an image copy.”

The example in Figure 165 requests the creation of a single output image copy data set for each area of the reloaded DEDB.
Enhancing performance

The following methods might significantly improve the performance of the reload function:

- output caching
- input thread processing
- using the SORT keyword
- using the SORT_OPTION keyword with the SORT keyword

Reload function output caching

Output caching with the OCACHE keyword during the reload function can reduce elapsed time for the function, at the cost of increased CPU utilization. A dataspace is used to implement the cache. The function has the same considerations as output caching for the change function. For more information, see “Enhancing performance” on page 163.

Reload function thread processing

Thread processing with the INPUT_THREADS keyword during the reload function can significantly speed processing. The function has the same considerations as thread processing for the unload function. For more information, see “Enhancing performance” on page 197.
Using SORT keyword with reload

You can use the SORT keyword to further improve performance of the reload process. This is beneficial for user-created load files or unload files that are in largely re-randomized sequence because of database changes.

- **SORT=NO** (the default) specifies that the load file is not to be sorted. The sequence of the records largely corresponds to the order in which the data will appear in the output area (UOW sequence).

- **SORT=AUTO** enables the reload function to determine whether sorting the sequential file will improve performance before execution of the RELOAD command. If it is determined that sorting the file will be beneficial, specifying AUTO causes a sort to be performed before the reload.

Using the file sort utility

For specific application processing needs, you can use the Fast Path/EP File Sort Utility to sort the unload file prior to application program execution. This command-driven utility provides a method for invoking your installation’s sort utility to perform a customized sort of the unload input file. In addition to providing other sorting alternatives, the File Sort utility provides a method of pre-sorting a file in RAP or load sequence.

For detailed information on using the File Sort Utility, see the “Supporting Utilities” chapter in the *Fast Path/EP Series Reference Manual*.

Using SORT_OPTION keyword with reload

The SORT_OPTION keyword provides a convenient method for supplying optional sort tuning parameters to your site’s Sort utility. SORT_OPTION=DYNALLOC (the default) specifies that sort work space will be dynamically allocated according to your Sort utility’s installation defaults.

For more information, see the reference manual for the sort product that is used at your site.
UNLOAD and RELOAD program extensions

The following program extensions accompany the unload/reload functions to support application programming interface with the functions:

- The Reload File Create utility (FABEUR6) can format and write database segments in any of three formats (HDUNLOAD, TFMT, or DBT).

- The Read Unloaded Database utility (FABEUR7) can retrieve database segments from a file that has been created by an unload function in any of three formats (HDUNLOAD, TFMT, or DBT).

For more information, see the Fast Path/EP Series Reference Manual.

Sample UNLOAD and RELOAD command scenarios

For scenarios that show how to use the UNLOAD and RELOAD commands with key related keywords and subcommands, see Appendix C, “Sample command scenarios.”
Extracting data from a DEDB

This chapter provides information on the capabilities, benefits, and use of the offline DEDB data extract function that is provided by Fast Path Analyzer/EP. This function lets you extract full or partial segment data from DEDBs while they are offline to IMS.

This chapter discusses how to specify the appropriate command syntax for all available extract functions. It also looks at the extract process from the user perspective as a series of chronological steps and explains the various results that can be obtained, depending on how you define the process in the job control language.

This chapter discusses the following topics:

Data extract function overview ................................................................. 236
  How extract works ............................................................................. 236
  Format flexibility .............................................................................. 236
  Benefits ............................................................................................. 237
Data extract function inputs and outputs .............................................. 237
DBRC considerations .......................................................................... 238
EXTRACT command keywords and subcommands ............................. 239
Process overview ............................................................................... 240
Selecting the extract file format ........................................................... 241
  EXTRACT_FORMAT=EXTRACT ......................................................... 242
  EXTRACT_FORMAT=HDUNLOAD ................................................... 243
  EXTRACT_FORMAT=USER ............................................................. 244
Selecting the database and areas ............................................................. 245
Allocating the input area data set ........................................................... 246
Allocating the extract output data set ..................................................... 247
  Creating a separate output data set for each area ......................... 247
  Combining multiple areas into a single data set ......................... 249
Selecting database segments for input and output ............................... 251
  Including and excluding segments for input ............................... 251
  Narrowing the selection ................................................................. 252
  Outputting segment records ........................................................... 255
  Samples of INCLUDE, EXCLUDE, and OUTPUT subcommands ...... 256
Controlling segment expansion ............................................................. 259
Controlling content of segment records ................................................. 259
**Data extract function overview**

The Fast Path Analyzer/EP data extract function provides an excellent alternative to user-written batch message processing programs (BMPs) for extracting segment data (full or partial segments) from DEDBs. The data extract function offers flexible, generalized segment selection and data extraction facilities. Selected segments and/or data items can be written to a sequential data set for subsequent processing by a user-written program.

**How extract works**

Fast Path Analyzer/EP completes a data extract that is based on how you combine the EXTRACT command with its associated keywords and subcommands. This command syntax defines the inputs to the extract process and the outputs from the extract process.

**Format flexibility**

The EXTRACT command is versatile and powerful. You can extract as much or as little data as you want from a DEDB. The format and content of the outcome (the extract output file) can be controlled by specifying a user-customized combination of command syntax.

**Input flexibility**

The data extract function lets you select the database and areas within the database to be used as input to the extract. You can use the DEDB or an image copy of a DEDB as input. The data extract function also lets you include or exclude certain segments for processing.
Output format flexibility

Extracted data is written to an output file by using one of three user-selectable output formats. The format that you select dictates the extent to which you can exercise control over the database-level and area-level header and trailer records and the content of the segment data records (including segment prefixes and suffixes).

- If the application that reads the extract data does not have specific formatting requirements, or if those requirements have not yet been determined, the default format (EXTRACT) should be used. A module is provided with Fast Path Analyzer/EP to facilitate reading the extracted data from a user-written program. For instructions about using this module, see the Fast Path/EP Series Reference Manual. For a sample COBOL program to retrieve extracted data, see the member PFUGXDR in the Fast Path/EP sample library.

- The data can be written in the same format (HDUNLOAD) as the Fast Path Reorg/EP unload function. This format should be used when the extracted data is used as input to the reload process.

- The data can be written in a custom-tailored format (USER). This format should be used when an existing application program requires extracted data in a specific format.

Benefits

The data extract function provides the following benefits:

- It increases productivity of application programmers by extracting data, using a control statement-driven, generalized, high-speed utility.

- It provides a convenient mechanism for creating test databases. Specified database records can be unloaded easily from a production database and reloaded to a test database.

- It eliminates IMS overhead because BMPs are not required.

Data extract function inputs and outputs

As shown in Figure 166, the data extract function extracts data from one or more area data sets of a DEDB (or an image copy of a DEDB) and writes the data to a sequential file.
DBRC considerations

When DBRC is active during execution of the extract function and the area is registered, the authorization level of the area data set is registered as shown in Table 17. If the area data set is used as input, the data set name must match the name that is registered with DBRC.

Table 17  DBRC settings for the extract function

<table>
<thead>
<tr>
<th>Command</th>
<th>Authorization</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXTRACT (area data set input)</td>
<td>read-with-integrity (RD)</td>
</tr>
<tr>
<td>EXTRACT (image copy input)</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Multiple area data sets (MADS) are not supported. The extract function searches the ADS (area data set) list that is registered for each area (in collating sequence by DD name). The product selects the first ADS that is marked as available for use and that has no error queue elements (EQEs). If an ADS meets both of these criteria, it is the only ADS that is used as input. All other area data sets are ignored.

If no area data set is marked as available, or if all available area data sets contain one or more EQEs, the extract function is not performed for that area.
**EXTRACT command keywords and subcommands**

Table 18 lists the command language that is used for each step in the data extract process. Some keywords are directly related to the EXTRACT command; others require an interim subcommand. To facilitate your understanding and correct coding of JCL to implement an extract, BMC recommends that you consider these steps in the order presented in Table 18.

**Table 18  EXTRACT command subcommands and keywords**

<table>
<thead>
<tr>
<th>Function</th>
<th>Command or subcommand</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>selecting the extract file format</td>
<td>EXTRACT</td>
<td>EXTRACT_FORMAT</td>
</tr>
<tr>
<td>selecting the database and areas</td>
<td>EXTRACT</td>
<td>DBD IAREA</td>
</tr>
<tr>
<td>dynamically allocating the extract input data set</td>
<td>EXTRACT</td>
<td>INPUT_DSN_MASK</td>
</tr>
<tr>
<td>dynamically allocating the extract output data set</td>
<td>OFILECTL</td>
<td>OUTPUT_DSN_MASK all associated keywords</td>
</tr>
<tr>
<td>selecting database segments for input</td>
<td>INCLUDE EXCLUDE</td>
<td>SAMPLE_INTERVAL SAMPLE_LIMIT SEGMENT WHERE</td>
</tr>
<tr>
<td>selecting database segments for output</td>
<td>OUTPUT</td>
<td>SEGMENT WHERE</td>
</tr>
<tr>
<td>controlling segment expansion</td>
<td>EXTRACT</td>
<td>EXPAND</td>
</tr>
<tr>
<td>controlling the content of segment records</td>
<td>OUTPUT</td>
<td>SEGMENT FIELDS</td>
</tr>
<tr>
<td>controlling the content of segment record prefixes and suffixes</td>
<td>EXTRACT</td>
<td>SEGMENT_RECORD_PREFIX SEGMENT_RECORD_SUFFIX</td>
</tr>
<tr>
<td>controlling header and trailer records</td>
<td>USER_RECORD</td>
<td>BREAK FIELDS</td>
</tr>
<tr>
<td>sorting the extract file</td>
<td>EXTRACT</td>
<td>SORT SORT_OPTION</td>
</tr>
<tr>
<td>enhancing performance</td>
<td>EXTRACT</td>
<td>ICACHE</td>
</tr>
</tbody>
</table>

For more information about the following topics, see the *Fast Path/EP Series Reference Manual*:

- syntax of commands, subcommands, and keywords that are discussed in this book
- diagrams that show the syntax and available parameters and values for Fast Path Offline Suite commands and subcommands
Process overview

Because the EXTRACT command makes only one pass through the data in the area, it is important to understand the chronological process that it executes. Subcommands and keywords are executed by the data extract function to control data content and record formatting at each step in the process.

Figure 167 shows how each step in the process is executed as specified by command language elements. However, because the availability of some options are dependent on other options, the subcommands and keywords are discussed in this chapter in the order that you should consider when becoming familiar with how to code a customized extract.

Figure 167  Extract data control process

<table>
<thead>
<tr>
<th>Subcommands/Keywords</th>
<th>Process Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>IAREA keyword</td>
<td>Area A</td>
</tr>
<tr>
<td>INCLUDE/EXCLUDE subcommands</td>
<td></td>
</tr>
<tr>
<td>SEGMENT keyword</td>
<td>Area B</td>
</tr>
<tr>
<td>WHERE keyword</td>
<td></td>
</tr>
<tr>
<td>SAMPLE_LIMIT keyword</td>
<td>segment selection</td>
</tr>
<tr>
<td>SAMPLE_INTERVAL keyword</td>
<td></td>
</tr>
<tr>
<td>OFILECTL subcommand and associated keywords</td>
<td>defining extract data set output</td>
</tr>
<tr>
<td>OUTPUT subcommand</td>
<td></td>
</tr>
<tr>
<td>SEGMENT keyword</td>
<td></td>
</tr>
<tr>
<td>FIELDS keyword</td>
<td></td>
</tr>
<tr>
<td>WHERE keyword</td>
<td></td>
</tr>
<tr>
<td>EXTRACT_FORMAT keyword</td>
<td></td>
</tr>
<tr>
<td>SEGMENT_RECORD_PREFIX keyword</td>
<td></td>
</tr>
<tr>
<td>SEGMENT_RECORD_SUFFIX keyword</td>
<td></td>
</tr>
<tr>
<td>USER_RECORD subcommand</td>
<td></td>
</tr>
<tr>
<td>BREAK keyword</td>
<td></td>
</tr>
<tr>
<td>FIELDS keyword</td>
<td></td>
</tr>
<tr>
<td>EXPAND keyword</td>
<td></td>
</tr>
<tr>
<td>SORT keyword, SORT_OPTION keyword</td>
<td></td>
</tr>
</tbody>
</table>
Selecting the extract file format

The general format of the extract output file can be controlled by specifying one of three values that are available for the EXTRACT_FORMAT keyword. As summarized in Table 19, each available choice enables a different set of more detailed options for tailoring extract output. Certain functions that affect record layout as well as the output data are available only for particular extract file formats.

<table>
<thead>
<tr>
<th>Function</th>
<th>EXTRACT</th>
<th>HDUNLOAD</th>
<th>USER</th>
</tr>
</thead>
<tbody>
<tr>
<td>selecting segments for input</td>
<td>INCLUDE</td>
<td>EXCLUDE</td>
<td>INCLUDE</td>
</tr>
<tr>
<td></td>
<td>SEGMENT WHERE</td>
<td>SEGMENT WHERE</td>
<td>SEGMENT WHERE</td>
</tr>
<tr>
<td></td>
<td>SAMPLE_LIMIT</td>
<td>SAMPLE_LIMIT</td>
<td>SAMPLE_LIMIT</td>
</tr>
<tr>
<td>outputting segment records</td>
<td>OUTPUT</td>
<td>SEGMENT * WHERE</td>
<td>OUTPUT SEGMENT WHERE</td>
</tr>
<tr>
<td>(controlling the generation of segment records to the output file)</td>
<td>INCLUDE</td>
<td>EXCLUDE</td>
<td>INCLUDE</td>
</tr>
<tr>
<td></td>
<td>SEGMENT WHERE</td>
<td>SEGMENT WHERE</td>
<td>SEGMENT WHERE</td>
</tr>
<tr>
<td>controlling the content of output segment records</td>
<td>OUTPUT</td>
<td>FIELDS WHERE</td>
<td>OUTPUT FIELDS WHERE</td>
</tr>
<tr>
<td>controlling content of the segment record prefix</td>
<td>no user control; standard EXTRACT format is used.</td>
<td>no user control; standard HDUNLOAD format is used</td>
<td>SEGMENT_RECORD_PREFIX</td>
</tr>
<tr>
<td>controlling content of the segment record suffix</td>
<td>no user control; no suffix is generated.</td>
<td>no user control; no suffix is generated</td>
<td>SEGMENT_RECORD_SUFFIX</td>
</tr>
<tr>
<td>controlling header records</td>
<td>no user control; standard EXTRACT header records are generated.</td>
<td>no user control; standard HDUNLOAD header records are generated</td>
<td>USER_RECORD BREAK FIELDS</td>
</tr>
<tr>
<td>controlling trailer records</td>
<td>no user control; no trailer records are generated.</td>
<td>no user control; standard HDUNLOAD trailer records are generated</td>
<td>USER_RECORD BREAK FIELDS</td>
</tr>
<tr>
<td>sorting the extract file</td>
<td>SORT</td>
<td>SORT_OPTION</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>SORT_OPTION</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Table 19 Data extract tailoring options according to value specified for EXTRACT_FORMAT keyword
EXTRACT_FORMAT=EXTRACT (the default) specifies that the file be formatted in the standard Fast Path Analyzer/EP extract file format. This format should be used when the application that will read the extract data does not have specific formatting requirements or when those requirements have not yet been determined. For the record layout that is used in this format, see the Fast Path/EP Series Reference Manual.

EXTRACT_FORMAT=EXTRACT lets you control the following processes:

- data expansion (EXPAND keyword)
- selection of segments for input (INCLUDE/EXCLUDE subcommands and associated keywords)
- exclusion of data records for selected segments from the output file (OUTPUT subcommand with optional WHERE keyword)
- controlling segment data content (OUTPUT subcommand with FIELDS keyword)
- sorting of the extract file (SORT and SORT_OPTION keywords)

EXTRACT_FORMAT=EXTRACT does not let you control the following processes:

- specifying a segment data record prefix or suffix
- specifying header or trailer records

Figure 168 shows the record layout when EXTRACT_FORMAT=EXTRACT is selected.

**Figure 168  Map of the EXTRACT format extract output file**

![Diagram of the EXTRACT format extract output file]
EXTRACT_FORMAT=HDUNLOAD specifies that the file be written in a format that is compatible with the IBM HD Reorganization Unload utility. This format is the same format that is used by the Fast Path Reorg/EP unload and reload functions and should be used when the extracted data is to be used as input to the reload process.

EXTRACT_FORMAT=HDUNLOAD lets you control the following processes:

- data expansion (EXPAND keyword)
- selection of segments for input (INCLUDE/EXCLUDE subcommands and associated keywords)
- segment data content (OUTPUT subcommand with FIELDS keyword)

EXTRACT_FORMAT=HDUNLOAD *does not* let you control the following processes:

- exclusion of data records for selected segments within a hierarchical path
- specifying a segment data record prefix or suffix
- specifying header or trailer records
- sorting of the extract file

Figure 169 shows the record layout when EXTRACT_FORMAT=HDUNLOAD is selected.

**Figure 169  Map of the HDUNLOAD format extract output file**
EXTRACT_FORMAT=USER

EXTRACT_FORMAT=USER allows the most detailed level of control of the output file. This format should be used when an existing application program requires extracted data in a specific format. EXTRACT_FORMAT=USER lets you control the following processes:

- data expansion (EXPAND keyword)
- selection of segments for input (INCLUDE/EXCLUDE subcommands and associated keywords)
- exclusion of segment data records for selected segments (OUTPUT subcommand with optional WHERE keyword)
- segment data content (OUTPUT subcommand with FIELDS keyword)
- segment data record prefix and suffix (SEGMENT_RECORD_PREFIX and SEGMENT_RECORD_SUFFIX keywords)
- header and trailer records (USER_RECORD subcommand and keywords)

EXTRACT_FORMAT=USER does not support sorting of the extract file.

Figure 170 shows the available user-selectable options for record layout when EXTRACT_FORMAT=USER is selected. Keywords that are used to customize record layouts are shown in boldface type, and are discussed in detail later in this chapter. All areas in the extract output file will have the same layout as specified by the various keywords.

Figure 170  Map of the USER format extract output file
Selecting the database and areas

The DBD keyword identifies the name of the DEDB (DBD name) from which data is to be extracted. This keyword is required when extracting in offline mode.

The IAREA keyword can be used to select specific areas from which the data is to be extracted. If you omit the IAREA keyword, data is extracted from all areas that are defined in the DEDB.

Areas can be specified on the IAREA keyword by using any combination of area names, area numbers, or a area ranges. The following parameters are available for the IAREA keyword:

- **IAREA=ALL** (default) or **IAREA=*** specifies all areas of the DEDB.
- **IAREA=areaname** specifies one or more areas by using the one-character to eight-character area name for each area specified. Multiple area names must be enclosed in parentheses and separated by commas.
- **IAREA=areanumber** specifies one or more areas by using the one-character to five-character area number for each area specified. Multiple area numbers must be enclosed in parentheses and separated by commas.
- **IAREA=(RANGE=(startarea,endarea))** specifies a consecutive range of areas using either **areaname** or **areanumber** parameters. The area number associated with **startarea** must be less than the area number associated with **endarea**.

An asterisk (*) can be used to specify all areas of the DEDB. When the * character is used with the RANGE keyword, it can be used to specify the beginning or ending range for specific areas of the DEDB.

To extract data from an entire DEDB, use a command set like the example shown in Figure 171.

*Figure 171  Sample control statement for extracting data from all areas*

```
EXTRACT DBD=dbdname,IAREA=ALL
```

To extract data only from specific areas, use a command set like the example shown in Figure 172.

*Figure 172  Sample control statement for extracting data from specific areas*

```
EXTRACT DBD=dbdname,IAREA=(AREANAM1,AREANAM3,RANGE=(5,8))
```
This control statement uses a combination of area names and area numbers to request that data is to be extracted from specific areas. Data will be extracted from the areas named AREANAM1 and AREANAM3. Also, data will be extracted from all consecutive areas from area number 5 to area number 8 (area5, area6, area7 and area8).

### Allocating the input area data set

For an offline extract, or when extracting data from an image copy, the area data set can be supplied in the JCL or can be accessed using dynamic allocation.

The areaname DD statement identifies the area data set to input to the extract function. The areaname DD statement can specify an image copy data set. If the areaname DD statement is omitted from the JCL, Fast Path Reorg/EP tries to dynamically allocate it.

The IMSACB DD statement identifies the ACB library containing the database definition that describes the area referenced by the areaname DD statement.

If you are using dynamic allocation, do not include the areaname DD statement. Fast Path Analyzer/EP tries to obtain the data set name for allocation in the following order:

1. If the INPUT_DSN_MASK keyword is specified, it is used to generate the data set name. The data set name can specify an image copy.

2. If DBRC is active and the area is registered, the registered area data set name is obtained from DBRC.

3. The STEPLIB is searched for the DFSMDA member that contains the data set name for this area.

If DBRC is active, the area is registered with DBRC, and the data set to be read is not an image copy, the allocated data set name must match the registered data set name regardless of how it is allocated.

Figure 173 shows an example using the INPUT_DSN_MASK keyword to generate the data set name for the extract. Specifying this keyword allocates VSAM area data set or image copy data set to be used as input to the extract.

**Figure 173  Allocating extract input using INPUT_DSN_MASK keyword**

```plaintext
EXTRACT DBD=PFPDBD1
INPUT_DSN_MASK='PFP.&AREA'
```
Allocating the extract output data set

The extract data sets can be allocated by using DD statements that are supplied in the JCL or by using dynamic allocation. BMC recommends dynamic allocation because of the flexibility and improved control that it provides.

By default, one output extract data set is produced for each input area. You can specify the optional OFILECTL subcommand to combine data that is obtained from one or more input areas into a single output data set. One output extract data set is created for each OFILECTL subcommand that is specified.

Regardless of which format you select for the extract output file, the DCB for each output file will always have the following attributes:

- RECFM=VB (variable block)
- LRECL= minimum record length for the file calculated automatically by Fast Path Analyzer/EP

You can, however, specify a larger value.

- BLKSIZE= calculated automatically by Fast Path Analyzer/EP

Creating a separate output data set for each area

If no OFILECTL subcommand is specified, a separate output extract data set is produced for each output area. The DD statement for each output extract data set is named OAREA.xxx, which uses a three-digit area number, or OARxxxxx, which uses a five-digit area number. The OAREA.xxx DD statement can only be used for area numbers 1 through 999. The OARxxxxx DD statement can be used for area numbers 1 through 2048, and must be used for area numbers greater than 999. If this DD statement is omitted from the JCL, Fast Path Analyzer/EP tries to dynamically allocate it by using the OUTPUT_DSN_MASK keyword. The data set name that is generated by using this mask must be a preexisting cataloged data set.

Using standard JCL

Use JCL DD statements named OAREA.xxx, where xxx is the three-digit area number that is listed in the DBD, or OARxxxxx, where xxxx is the five-digit area number that is listed in the DBD, to indicate the output extract data set that is produced for each input area.
Creating a separate output data set for each area

The example in Figure 174 requests the creation of one output extract data set for each input area. The data that is extracted from input area \texttt{areanam1} is written to the OAREA001 DD statement, the data that is extracted from input area \texttt{areanam2} is written to OAR00002, the data that is extracted from input area \texttt{areanam3} is written to OAREA003, and the data that is extracted from input area \texttt{area2048} is written to OAR02048.

**Figure 174  JCL to allocate extract data sets**

```plaintext
//PFP EXEC PGM=PFPMAIN,REGION=0M,
//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
//area1 DD DISP=SHR,DSN=IMS.RESLIB
//areanam1 DD DSN=area.dataset1,DISP=OLD
//areanam2 DD DSN=area.dataset2,DISP=OLD
//areanam3 DD DSN=area.dataset3,DISP=OLD
//area2048 DD DSN=area.data2048,DISP=OLD
//OAREA001 DD UNIT=TAPE,DISP=(NEW,CATLG),
//DSN=your.dataset.name1
//OAR00002 DD UNIT=TAPE,DISP=(NEW,CATLG),
//DSN=your.dataset.name2
//OAREA003 DD UNIT=TAPE,DISP=(NEW,CATLG),
//DSN=your.dataset.name3
//OAR02048 DD UNIT=TAPE,DISP=(NEW,CATLG),
//DSN=your.dataset.name2048
//PFPSYSIN DD *
  EXTRACT DBD=dbdname,
    IAREA=(areanam1,areanam2,areanam3,area2048)
/*
```

Using dynamic allocation

To use dynamic allocation, omit the OAREA\texttt{xxx} and OAR\texttt{xxxx} DD statements from the JCL. Use the OUTPUT_DSN_MASK keyword to supply a mask to construct the \textit{pre-existing} cataloged data set name. Substitution variables within the mask can be used to generate unique names for each output data set.

You can refer to a relative generation of a generation data group (GDG) by including it in the mask, such as in \texttt{gdg-name(-1)}. The relative generation number that you specify must be zero or less. If you refer to a GDG name without specifying a relative generation number, the product automatically appends a relative generation of zero.

The example in Figure 175 requests that a separate preallocated output extract data set be written for each input area.

**Figure 175  Dynamic allocation of extract data sets by using OUTPUT_DSN_MASK keyword**

```plaintext
EXTRACT DBD=dbdname,
  OUTPUT_DSN_MASK='dataset-name-mask'
```
Combining multiple areas into a single data set

Under certain circumstances, consolidation of extracted data from multiple inputs into one extract file can be convenient. You can specify the optional OFILECTL subcommand to combine the data that is obtained from one or more input areas into a single output file. One output extract data set is produced for each OFILECTL subcommand that is specified. Use the OAREA keyword on the OFILECTL subcommand to specify the areas that will be placed into the output extract data set.

**NOTE**

Each input area that is specified by using the IAREA keyword on the EXTRACT command must also be specified in the OAREA keyword of one (and only one) OFILECTL subcommand.

**Using OFILECTL with standard JCL**

Use the DDNAME keyword (*not* the DSNAME keyword) to indicate that the extract data set is to be allocated by using JCL DD statements.

The example in Figure 176 uses JCL to allocate an extract data set. Extracted data which is obtained from all areas that are defined in the database is combined and written into a single extract data set.

**Figure 176  JCL to allocate an extract data set by using the DDNAME keyword**

```
//PFP EXEC  PGM=PFPMAIN,REGION=0M.
//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
// ddname1 DD UNIT=TAPE,DISP=(NEW,CATLG),
//         DSN=your.dataset.name
//PFPSYSIN DD *
EXTRACT DBD=dbdname
  OFILECTL DDNAME=ddname1
/*
```

**Using OFILECTL with dynamic allocation**

Use the DSNAME keyword (*not* the DDNAME keyword) to indicate that the extract data set is to be dynamically allocated. Use the DSNAME keyword to supply a mask to construct the data set name. Substitution variables within the mask can be used to generate unique names for each output data set.

Additional related keywords such as UNIT, DISP, and DATACLAS can be used to control the allocation and disposition of the data set, similar to the corresponding JCL keywords.
The example in Figure 177 requests the creation of two extract data sets. The data extracted from input areas *areanam1* and *areanam2* is written into one extract data set, and the data from *areanam3* is written into a separate data set. Both output files are written to tape devices and are recorded in the system catalog on completion.

**Figure 177  Dynamic allocation of extract data sets by using DSNAME keyword**

```plaintext
EXTRACT DBD=dbdname,IAREA=(areanam1,areanam2,areanam3)
  OFILECTL OAREA=(areanam1,areanam2),
    DSNAME='dataset-name1-mask',
    UNIT=TAPE,DISP=(NEW,CATLG)
  OFILECTL OAREA=(areanam3),
    DSNAME='dataset-name2-mask',
    UNIT=TAPE,DISP=(NEW,CATLG)
```

**NOTE**

The ddname that is used for dynamic allocation of the extract output data sets depends on the area number of the first area that is listed in the OAREA keyword. If the first area that is listed in the OAREA keyword has an area number between 1 and 999, the OAREAaaa DD is used for dynamic allocation, where *aaa* is the three-digit area number. If the first area that is listed in the OAREA keyword has an area number between 1000 and 2048, OARxxxx is used for dynamic allocation, where *xxxx* is the area number. If an OAREAxxx or OARxxxx DD statement is supplied in the JCL, it is used instead of the dynamic allocation parameters.

The data set name that is specified with the DSNAME keyword can be used to construct a GDG by specifying the base name for the GDG. As an option, you can include a relative generation number as part of the DSNAME keyword.

If the data set name matches an existing GDG base name, the product automatically appends the appropriate generation information to the base name that is equivalent to relative generation (+1), if not specified explicitly. DISP=NEW is required when a new generation data set is created.

The example in Figure 178 requests that all extract data be written to a single data set. The output data set, specified by using the DSNAME keyword, is allocated on disk and is a new generation within a GDG.

**Figure 178  Using a generation data set group**

```plaintext
EXTRACT DBD=dbdname
  OFILECTL DSNAME='gdg-dataset-name-mask(+1)',
    DISP=(NEW,CATLG),UNIT=3390,VOLSER=123456,
    SPACE=(CYL,100,10,RLSE)
```
Selecting database segments for input and output

By default, all root segments and their dependent segments (including SDEP segments) are selected. You can modify the selection of segments by using the optional INCLUDE and EXCLUDE subcommands.

The segments that you include (or do not exclude) by using these subcommands are selected and forwarded to the output processing for record formatting. You can also use several keywords with the INCLUDE and EXCLUDE subcommands to further refine the input selection process.

The OUTPUT subcommand specifies only segments that you want written to the output extract file. You can also use selected keywords on the OUTPUT subcommand to apply conditional selection criteria. You cannot, however, specify segments on an OUTPUT subcommand that were excluded from your initial selection with an EXCLUDE subcommand or a conditional INCLUDE subcommand.

Including and excluding segments for input

The INCLUDE subcommand lets you specify criteria for selecting segments of interest for the extract. If the INCLUDE subcommand selection criteria are met, the selected segments and their dependents are forwarded for output processing.

The EXCLUDE subcommand lets you specify criteria for excluding segments from the extract. If the EXCLUDE subcommand selection criteria are met, the selected segments and their dependents are not forwarded for output processing.

**NOTE**

You cannot include a dependent segment and exclude its parent segment. Specifying a parent segment on the EXCLUDE subcommand also excludes all dependent segments.

The SEGMENT keyword specifies the name of the segment to which the selection criteria are applied. Additional selection criteria can be specified by using the WHERE keyword, which uses field names or positional identifiers with relational or Boolean logical expressions.

A simple INCLUDE or EXCLUDE subcommand uses the required SEGMENT keyword *without* additional keywords. Understanding the following rules that apply to the basic subcommands is prerequisite to using additional keywords with INCLUDE and EXCLUDE:
Narrowing the selection

- By default, if no INCLUDE or EXCLUDE statement is used with the EXTRACT command, all segments are selected.

- The INCLUDE subcommand selects the segment that is identified by the SEGMENT keyword and all dependents of the segment. Including a segment in this manner also selects the parent hierarchy chain above the segment.

- The EXCLUDE subcommand excludes the segment that is identified by the SEGMENT keyword and all dependents of the segment. You cannot include a segment and exclude a root or a parent of the segment. Excluding a parent excludes the entire hierarchy beneath the parent.

A basic EXTRACT command with an INCLUDE and EXCLUDE subcommand is shown in Figure 179.

Figure 179 Using the EXTRACT command with INCLUDE and EXCLUDE subcommands

```
EXTRACT IAREA=(area1,area3)
INCLUDE SEGMENT=segname1
EXCLUDE SEGMENT=segname2
```

Narrowing the selection

While identifying the segments of interest for the extract by using the INCLUDE and EXCLUDE subcommands, you can apply additional criteria to narrow down the selection by using the following keywords:

- The WHERE keyword selects segments by applying conditional criteria that is specified in a Boolean expression.

- The SAMPLE_INTERVAL keyword selects segments by intervals of occurrence.

- The SAMPLE_LIMIT keyword sets a limit on the number of segments meeting the criteria that will be selected.

Applying Conditional Criteria

You can also narrow segment selection by using the WHERE keyword. This combination applies relational and Boolean operators to apply conditional criteria when selecting segments. For information about the operations and syntax of the WHERE keyword and syntax rules for coding these expressions, see the Fast Path/EP Series Reference Manual.
In the example in Figure 180, four segments exist in the database but only two segments (PFPROOT and PFPADDR) are written to the two extracted files. PFPACCT and its dependents are explicitly excluded by the EXCLUDE subcommand. Only the PFPROOT segments that have field FLDNAME equal BMCPFP and a value greater than 10,000 in position 46 are written to the output file. Only the PFPADDR segments (of the selected PFPROOTS), where position 23 is equal to 22, are written.

**Figure 180  EXTRACT Command using INCLUDE, EXCLUDE, and WHERE**

```plaintext
EXTRACT IAREA=(area1, area2)
    INCLUDE SEGMENT=PFPROOT,
        WHERE=(FLDNAME EQ C'BMCPFP' AND 46:5P GT P'10000')
    INCLUDE SEGMENT=PFPADDR,
        WHERE=(23:1X EQ X'22')
    EXCLUDE SEGMENT=PFPACCT
```

**Selecting specific database records by root key**

When the selection criteria for the extract identifies a list of root key values, the offline extract process will invoke the randomizer to retrieve these root segments directly. Performance is improved significantly because a sequential read of all root addressable pointers (RAPs) within the area is not necessary to select segments that meet the criteria.

The EXTRACT command will examine the selection criteria specified for the root segment to determine if it identifies root key values for all segments that match the selection criteria. If this condition is met, the randomizer will be used to read the root segments directly.

Root key values are specified by using the WHERE keyword on the INCLUDE subcommand for the root segment. The WHERE keyword must specify a special class of expressions called a root key expression that identifies every root key for the segments requested for inclusion.

The offline extract process invokes the randomizer to select root segments under the following conditions:

- The WHERE keyword must include a root key expression that identifies the root key with a logical EQ and a literal.

- The root key can be specified by using its segment search key field as defined in the DBD or by precisely specifying its position and length (with a column:length designation) and a data type.

- The root key expression on the WHERE keyword can be followed by other selection criteria by using a logical AND.
The root key expression on the WHERE keyword can include multiple root key expression values by using a logical OR.

Multiple INCLUDE subcommands can be specified under a single EXTRACT command. However, a root key expression must be specified on the WHERE keyword under each INCLUDE subcommand.

The example in Figure 181 requests an offline extract for a database containing six segments. Because the EXTRACT command includes three separate INCLUDE subcommands, PFPROOT segments (and their associated dependent segments) that meet any of the following criteria are written to the output file:

- field named ROOTKEY that is equal to BMCPFP
- 6-byte root key field beginning in column 3 that is equal to PFPX4Z
- field named ROOTKEY that is equal to PFP500 and 3-byte field beginning in column 40 that is equal to 123

Because each WHERE keyword in Figure 181 specifies a valid root key expression, the randomizer will be invoked to improve performance of the offline extract.

**Figure 181  EXTRACT command using INCLUDE statements that identify root keys and invoke the randomizer**

```
EXTRACT IAREA=( areanaml, areanam3)
   INCLUDE SEGMENT=PFPROOT,
       WHERE=(ROOTKEY EQ C'BMCPFP')
   INCLUDE SEGMENT=PFPROOT,
       WHERE=(3:6 EQ C'PFPX4Z')
   INCLUDE SEGMENT=PFPROOT,
       WHERE=(ROOTKEY EQ C'PFP500' AND 40:3 EQ '123')
```

**Using criteria that are applied to alternate segments**

You can specify segment selection, based on criteria that are applied to alternate segments, by using a segment-qualified field reference within the WHERE keyword. Only the parentage segments within the hierarchical path from the root can be used for qualification.

In the example in Figure 182, SEGC is selected for extract only if FIELD1 in SEGB is equal to the character string BMC.

**Figure 182  Qualifying field references using WHERE keyword**

```
EXTRACT IAREA=( areanaml, areanam3)
   INCLUDE SEGMENT=SEGC,
       WHERE=(SEGB.FIELD1 EQ C'BMC')
```
### Segment sampling

The SAMPLE_INTERVAL keyword can be used to specify segment selection based on the frequency of the segment occurrence. For example, if you have specified SAMPLE_INTERVAL=5, every fifth segment will be selected. For a root segment, every fifth occurrence in the database and all of its dependents will be selected. For a dependent segment, every fifth occurrence under its parent is selected.

The SAMPLE_LIMIT keyword can be used to specify the maximum number of segment occurrences to be selected. For example, if you have specified SAMPLE_LIMIT=500, the first 500 segments are selected. For a root segment, the first 500 occurrences and all its dependents are selected. For a dependent segment, the first 500 occurrences within its parent are selected.

In the example in Figure 183, only one area is input, so only one output extract file is created. The output extract file is sorted by root key. Every ninth occurrence of segment PFPROOT and all its dependents are selected, with a maximum of 5000 occurrences of PFPROOT.

#### Figure 183  EXTRACT command with SAMPLE_INTERVAL and SAMPLE_LIMIT keywords

```plaintext
EXTRACT IAREA=areanam2,
  EXTRACT_FORMAT=EXTRACT,
  SORT=YES
INCLUDE SEGMENT=PFPROOT,
  SAMPLE_INTERVAL=9,
  SAMPLE_LIMIT=5000
EXCLUDE SEGMENT=PFPADDR,
  SAMPLE_LIMIT=5
```

The EXCLUDE subcommand changes the result of the previous INCLUDE for dependent segment PFPADDR. For each PFPROOT segment, the first five segment occurrences for segment PFPADDR are omitted. If, for example, PFPADDR has eight occurrences, the last three segment occurrences would be selected. If segment PFPADDR had only four occurrences under another PFPROOT, no PFPADDR segments (or their dependents) would be selected.

### Outputting segment records

The optional OUTPUT subcommand can be used to limit the writing of segment records to the output file. By default, if no OUTPUT subcommands are specified, one segment record is written for each selected segment.

When one or more OUTPUT subcommands are specified, segment records are written only for segments that are specified. No output record is written for segment types that are not specified on any OUTPUT subcommand.
You can use the OUTPUT subcommand to specify only segments that you want written to the output extract file. You must include a separate OUTPUT subcommand for each segment type that you want included in the output, as shown in Figure 184.

Figure 184  Using the OUTPUT subcommand to control generation of segment records

```
EXTRACT DBD=dbdname, IAREA=ALL
    OUTPUT SEGMENT=PFPROOT
```

The writing of the output segment record can be controlled by using conditional qualification that is specified by the optional WHERE keyword on the OUTPUT subcommand. If multiple OUTPUT subcommands are specified for the same segment, the first subcommand in which the selection criteria is satisfied will be used. If the selection criteria is not satisfied for any OUTPUT subcommand, no output record is written for the segment.

Figure 185 shows an example where conditional qualification is used for a segment record during an data extract. If the conditional qualification that is specified by the WHERE keyword is met, the segment record is output. If the conditional qualification is not met, the segment record is not output.

Figure 185  Using the WHERE keyword to conditionally specify segment output

```
EXTRACT DBD=dbdname, IAREA=ALL
    OUTPUT SEGMENT=PFPROOT,
        WHERE=(23:3 EQ C’’)
```

Samples of INCLUDE, EXCLUDE, and OUTPUT subcommands

Figure 186 shows the root segment SEGA and its hierarchy. Using this hierarchy as a model, the tables in this section show examples of the EXTRACT command combined with the INCLUDE, EXCLUDE, and OUTPUT subcommands and the resulting segments that are selected (by INCLUDE and EXCLUDE subcommands) and output (by OUTPUT subcommand).
Two separate tables are presented to show different segment output results with the HDUNLOAD format because of its segment hierarchical path restrictions:

- **Table 20** shows selected segments and segment output when `EXTRACT_FORMAT=EXTRACT` or `EXTRACT_FORMAT=USER` has been specified.

- **Table 21** shows selected segments and segment output when `EXTRACT_FORMAT=HDUNLOAD` has been specified.

### Table 20  Using INCLUDE, EXCLUDE and OUTPUT subcommands with the EXTRACT command and `EXTRACT_FORMAT=EXTRACT` or `EXTRACT_FORMAT=USER` (part 1 of 2)

<table>
<thead>
<tr>
<th>Command set</th>
<th>Selected segments</th>
<th>Output segments</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXTRACT IAREA=ALL No INCLUDE, EXCLUDE, or OUTPUT subcommands are specified with the EXTRACT command.</td>
<td>All roots and their dependent segments are selected.</td>
<td>A segment record is written for each selected segment.</td>
</tr>
<tr>
<td>EXCLUDE SEGMENT=SEGA</td>
<td>No segments are selected.</td>
<td>No segment records are output.</td>
</tr>
<tr>
<td>EXCLUDE SEGMENT= SEGA INCLUDE SEGMENT=SEGB</td>
<td>No segments are selected. (Excluding a parent segment excludes all of its dependent segments.)</td>
<td>No segment records are output.</td>
</tr>
<tr>
<td>EXCLUDE SEGMENT=SEGB</td>
<td>SEGA, SDEP, SEGE, SEGF, and SEGG are selected.</td>
<td>A segment record is written for each selected segment.</td>
</tr>
<tr>
<td>INCLUDE SEGMENT=SEGA, WHERE=(selection criteria)</td>
<td>All SEGA segments that meet criteria and their dependents are selected.</td>
<td>A segment record is written for each selected segment.</td>
</tr>
<tr>
<td>EXCLUDE SEGMENT=SDEP EXCLUDE SEGMENT=SEGB EXCLUDE SEGMENT=SEGE OUTPUT SEGMENT=SEGG</td>
<td>SEGA and SEGG are selected.</td>
<td>A segment record is written for each SEG that is selected. (This command set is equivalent to the command set shown for SEGG.)</td>
</tr>
</tbody>
</table>
Samples of INCLUDE, EXCLUDE, and OUTPUT subcommands

Table 20  Using INCLUDE, EXCLUDE and OUTPUT subcommands with the EXTRACT command and EXTRACT_FORMAT=EXTRACT or EXTRACT_FORMAT=USER (part 2 of 2)

<table>
<thead>
<tr>
<th>Command set</th>
<th>Selected segments</th>
<th>Output segments</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPUT SEGMENT=SEGG</td>
<td>All roots and their dependent segments are selected.</td>
<td>A segment record is written for each SEGG selected.</td>
</tr>
<tr>
<td>INCLUDE SEGMENT=SEGB, WHERE=(selection criteria) EXCLUDE SEGMENT=SEGC OUTPUT SEGMENT=SEGD, WHERE=(selection criteria)</td>
<td>All SEGA segments are selected. SEGB segments that meet the selection criteria in the WHERE keyword are selected if they are children of a selected SEGB segment.</td>
<td>A segment record is written for each selected SEGD segment that meets the criteria on the OUTPUT subcommand.</td>
</tr>
</tbody>
</table>

Table 21  Using INCLUDE, EXCLUDE and OUTPUT subcommands with the EXTRACT command and EXTRACT_FORMAT=HDUNLOAD

<table>
<thead>
<tr>
<th>Command set</th>
<th>Selected segments</th>
<th>Output segments</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXTRACT IAREA=ALL No INCLUDE, EXCLUDE, or OUTPUT subcommands are specified with the EXTRACT command.</td>
<td>All roots and their dependent segments are selected.</td>
<td>A segment record is written for each selected segment.</td>
</tr>
<tr>
<td>EXCLUDE SEGMENT=SEGA</td>
<td>No segments are selected.</td>
<td>No segment records are output.</td>
</tr>
<tr>
<td>EXCLUDE SEGMENT= SEGA INCLUDE SEGMENT=SEGB</td>
<td>No segments are selected. (Excluding a parent segment excludes all of its dependent segments.)</td>
<td>No segment records are output.</td>
</tr>
<tr>
<td>EXCLUDE SEGMENT=SEGB</td>
<td>SEGA, SDEP, SEGE, SEGF, and SEGG are selected.</td>
<td>A segment record is written for each selected segment.</td>
</tr>
<tr>
<td>INCLUDE SEGMENT=SEGA, WHERE=(selection criteria)</td>
<td>All SEGA segments that meet criteria and their dependents are selected.</td>
<td>A segment record is written for each selected segment.</td>
</tr>
<tr>
<td>EXCLUDE SEGMENT=SDEP EXCLUDE SEGMENT=SEGB EXCLUDE SEGMENT=SEGE OUTPUT SEGMENT=SEGG</td>
<td>SEGA and SEGG are selected.</td>
<td>All SEGA and SEGG segments selected are written.</td>
</tr>
<tr>
<td>OUTPUT SEGMENT=SEGG</td>
<td>All roots and their dependent segments are selected.</td>
<td>All segments are written.</td>
</tr>
<tr>
<td>INCLUDE SEGMENT=SEGB, WHERE=(selection criteria) EXCLUDE SEGMENT=SEGC OUTPUT SEGMENT=SEGD, WHERE=(selection criteria)</td>
<td>All SEGA, SDEP, SEGE, SEGF, and SEGG segments are selected. Only SEGB (and SEGD children) segments that meet the INCLUDE selection criteria are selected.</td>
<td>All SEGA, SDEP, SEGE, SEGF, and SEGG segments are written. All SEGB (and SEGD children) segments that meet the selection criteria are written.</td>
</tr>
</tbody>
</table>
Controlling segment expansion

The output file can contain data from compressed segments. As an option, the extracted data can be expanded by using the `EXPAND` keyword. This keyword can be used, regardless of the value that you specified for the `EXTRACT_FORMAT` keyword:

- `EXPAND=NO` (the default) specifies that the segment data be written to the extract output file unchanged.
- `EXPAND=YES` specifies that compressed segment data be expanded when written to the extract output file.
- `EXPAND=(segment1, segment2, ..., segmentn)` causes only the compressed segments that are listed by name to be expanded when written to the extract output file.

**NOTE**

If the `FIELDS` keyword is used with the `OUTPUT` subcommand to refer to the content of a compressed segment, you must expand the segment by using the `EXPAND=YES` value or the `EXPAND=(segment1, segment2, ..., segmentn)` syntax on the `EXTRACT` command. This expansion will ensure that the operands which are specified on the `FIELDS` keyword correspond to the proper segment columns.

**Figure 187** shows an example of using the `EXPAND` keyword to selectively specify segment expansion. Only `segment1` and `segment3` in `area1` are expanded; all other segments are written to the extract output file unchanged.

**Controlling content of segment records**

The `OUTPUT` subcommand and its associated keywords can be used to control content of the segment record. By default, the output record will contain the full, unmodified contents of the selected segment occurrence.

If you do not want to write the full segment contents, you can identify individual fields to be written to the extract output file by using the `FIELDS` keyword. The `FIELDS` keyword specifies a list of expressions; the value of each expression is evaluated and placed into the output record in the order that the expressions are specified. For details about the operations and syntax of the `FIELDS` keyword and syntax rules for coding expressions, see the *Fast Path/EP Series Reference Manual.*
Controlling content of segment records

### WARNING

When EXTRACT_FORMAT=HDUNLOAD is specified and the FIELDS keyword is used with the OUTPUT subcommand to change the key of a dependent segment, the segments must remain in ascending key sequence under the root key. Fast Path Analyzer/EP does not have a facility to resequence the dependent segments in the extract file. If the change to a dependent segment key is not specified in ascending key sequence, a failure occurs if the extracted data is used as input to a reload process.

### NOTE

When the FIELDS keyword is used with the OUTPUT subcommand to refer to the content of a compressed segment, you must expand the segment by using the EXPAND=YES value or the EXPAND=(segment1,segment2,...,segmentn) syntax on the EXTRACT command. This expansion will ensure that the operands which are specified on the FIELDS keyword correspond to the proper segment columns.

The example in Figure 188 shows the FIELDS keyword coded on the OUTPUT subcommand.

**Figure 188  EXTRACT command with OUTPUT subcommand and FIELDS keyword**

```
EXTRACT DBD=dbdname, IAREA=ALL
OUTPUT SEGMENT=PFPROOT,
    FIELDS=(3:5,40:9,ROOTFLD6)
```

In the example shown in Figure 188, only the root segments in the database are extracted to the output files. Output records for segment PFPROOT are modified, and only three fields are written to the output file. The first field starts in position 3 of PFPROOT and extends five bytes. The second field starts in position 40 of PFPROOT and extends nine bytes. The third field is defined as ROOTFLD6 in the DBD definition.

The content of the output segment record can be controlled by using conditional qualification that is specified by the optional WHERE keyword on the OUTPUT subcommand. By using the optional WHERE keyword on the OUTPUT subcommand, generation of an output record for a segment can be controlled by using conditional qualification. If multiple OUTPUT subcommands are specified for the same segment, the first subcommand in which the selection criteria is satisfied is used. If the selection criteria is not satisfied for any OUTPUT subcommand, no output record is written for the segment.

Figure 189 shows an example where an alternative format for the segment record is selected by using conditional qualification during an data extract. If the conditional qualification is met, the first OUTPUT subcommand is used and the FIELDS keyword generates a modified segment image. If the conditional qualification is not met, the second OUTPUT subcommand is selected and the unmodified segment is written.
Controlling content of segment record prefixes and suffixes

When you select EXTRACT_FORMAT=USER as the general extract formatting option, you can also control the content of the segment prefix and suffix by using the SEGMENT_RECORD_PREFIX and SEGMENT_RECORD_SUFFIX keywords.

The SEGMENT_RECORD_PREFIX keyword specifies a list of expressions. The value of each expression is evaluated and placed into the prefix portion of the segment output record in the order that the expressions are specified. If the SEGMENT_RECORD_PREFIX keyword is omitted, the output record contains no data (zero bytes) in the prefix when EXTRACT_FORMAT=USER has been specified.

The SEGMENT_RECORD_SUFFIX keyword specifies a list of expressions. The value of each expression is evaluated and placed into the suffix portion of the segment output record in the order that the expressions are specified. If the SEGMENT_RECORD_SUFFIX keyword is omitted, the output record contains no data (zero bytes) in the suffix when EXTRACT_FORMAT=USER has been specified.

For details about the operations and syntax of the SEGMENT_RECORD_PREFIX and SEGMENT_RECORD_SUFFIX keywords and syntax rules for coding expressions, see the Fast Path/EP Series Reference Manual.

The example in Figure 190 uses the SEGMENT_RECORD_PREFIX keyword to generate a simple prefix for the output segment records. The prefix will contain the segment name and its concatenated key value.

Figure 190  Using SEGMENT_RECORD_PREFIX keyword to control content of segment record prefix

| EXTRACT DBD=dbdname, IAREA=ALL, EXTRACT_FORMAT=USER,     |
| SEGMENT_RECORD_PREFIX=(SEGMENT_NAME,SEGMENT_CKEY) |

The example in Figure 191 uses the SEGMENT_RECORD_SUFFIX keyword to generate a simple suffix for the output segment records. The suffix will contain the segment code and its hierarchical level.

Figure 191  Using the WHERE keyword to conditionally specify segment content

| EXTRACT DBD=dbdname, IAREA=ALL, OUTPUT SEGMENT=PFPROOT, |
| FIELDS=(3:20,C'ABC',23:*), WHERE=(23:3 EQ C' '), |
| OUTPUT SEGMENT=PFPROOT |
Controlling header and trailer records

When you select EXTRACT_FORMAT=USER as the general extract formatting option, you can also control the content of the segment header and trailer records by using the optional USER_RECORD subcommand and its associated BREAK and FIELDS keywords.

Specifying the generation of header and trailer records

The value that is specified for the BREAK keyword defines a trigger event that causes the record to be written. Table 22 shows the four options that available for the BREAK keyword, which correspond to the types of header and trailer records that can be generated.

<table>
<thead>
<tr>
<th>Value for BREAK keyword</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>BREAK=(DATABASE, BEFORE)</td>
<td>database-wide header record</td>
</tr>
<tr>
<td>BREAK=(DATABASE, AFTER)</td>
<td>database-wide trailer record</td>
</tr>
<tr>
<td>BREAK=(AREA, BEFORE)</td>
<td>area header record</td>
</tr>
<tr>
<td>BREAK=(AREA, AFTER)</td>
<td>area trailer record</td>
</tr>
</tbody>
</table>

One output record is generated for each occurrence of the USER_RECORD subcommand. When multiple subcommands with the same BREAK trigger are specified, one record is generated for each subcommand in the same order that the subcommands are specified.

Defining content of header and trailer records

The values that are specified for the FIELDS keyword define the content of the record. The value of each expression is placed into the output record in the order specified. For details about the operations and syntax of the FIELDS keyword, see the Fast Path/EP Series Reference Manual. For detailed syntax rules for coding expressions, see the Fast Path/EP Series Reference Manual.
The example in Figure 192 uses BREAK and FIELDS keywords with the USER_RECORD subcommand to create a database–wide header record and an area trailer record.

### Figure 192 Using BREAK and FIELDS keywords to create database header and trailer records

```sql
EXTRACT DBD=dbdname, IAREA=ALL,EXTRACT_FORMAT=USER
USER_RECORD BREAK=(DATABASE,BEFORE),
    FIELDS=(DBD_NAME,DBD_AREAS)
USER_RECORD BREAK=(AREA,AFTER),
    FIELDS=(AREA_NAME,SEGMENT_COUNT())
```

## Sorting the extract file

When you select EXTRACT_FORMAT=EXTRACT as the general extract formatting option, you can also sort the extract file. The following options are available for sorting the extract file:

- using the SORT keyword
- using the SORT_OPTION keyword

### Using the SORT keyword

The output file that is created by the data extract function can be sorted into root key sequence. The sequence is specified by using the SORT keyword. A sort key is appended to the beginning of the record. The following values are available:

- **SORT=NO** (the default) specifies that the output file not be sorted. The sequence of the records corresponds to the order in which the data appears in the input area (RAP sequence).

- **SORT=YES** specifies that the output file be sorted. The sequence of the records corresponds to the symbolic key of the root segments.

When the output file is sorted, the layout of the data records is changed. When sorting extract output files, Fast Path Analyzer/EP interacts with the sort product that is used at your site to pass appropriate sort work space and sort message information. The sorted and unsorted record layouts are shown in the *Fast Path/EP Series Reference Manual*. 


Using the SORT_OPTION keyword

The SORT_OPTION keyword provides a convenient method for supplying optional sort tuning parameters to your site’s Sort utility.

SORT_OPTION=DYNALLOC (the default) specifies that sort work space will be dynamically allocated according to your Sort utility’s installation defaults.

For more information, see the reference manual for the sort product that is used at your site.

Enhancing performance by input caching

The effect of caching on the input data set depends on the following factors:

- type of input (image copy or area data set)
- type of caching that is used
- how SDEP segments (if any) are processed

Input caching using the ICACHE keyword can speed processing significantly. A dataspace is used to implement the cache.

If an image copy is used as input, the extract function automatically uses caching of IOVF and SDEP control intervals (CIs). If resources are not available to support the required IOVF and/or SDEP caching, the extract function terminates.

If an area data set is used as input, input caching is optional. If resources are available, specify IOVF input caching. If the SDEP segment data is being extracted and resources are available, also specify SDEP caching.

Sample EXTRACT command scenarios

For scenarios that show how to use the EXTRACT command with key related keywords and subcommands, see Appendix C, “Sample command scenarios.”
Chapter 8 Initializing a DEDB

This chapter provides information on the initialization function that is provided by Fast Path Reorg/EP. This function lets you format (initialize) a VSAM cluster that will be used for a DEDB. This chapter discusses the following topics:

Initialization function overview .................................................. 265
Initialization restrictions ......................................................... 266
Initialization function inputs and outputs ..................................... 266
DBRC considerations .............................................................. 267
INITIALIZE command keywords and subcommands ......................... 268
Selecting the database and areas ............................................... 268
Allocating the area data sets .................................................... 269
  Using DD statements ......................................................... 269
  Using dynamic allocation to name the area data set ....................... 270
  Using the ALLOCATE subcommand to customize dynamically defined areas 270
  ALLOCATE subcommand sample scenarios ................................ 273
Initializing associated indexes ................................................ 274
Producing an image copy during initialization ............................... 274
Enhancing performance ......................................................... 274
Sample INITIALIZE command scenarios .................................. 275

Initialization function overview

The Fast Path/EP initialization function lets you initialize a VSAM cluster that will be used for a DEDB in the same manner as the IBM DEDB Initialization Utility. A database must have allocated space and must be empty before you invoke the INITIALIZE command. The initialization function can be performed offline only.

Fast Path/EP change and reload functions automatically initialize the area (if required) before writing any segments to it. If a failure occurs after initialization is complete, the area still is initialized. You do not have to restart processing at the initialization phase.
Initialization restrictions

If SMS-managed storage classes are used for initialized areas and the defined cluster spans the area across multiple volumes, you must allocate the area to a guaranteed space storage class. The initialization function will not format any portion of the area that resides on a candidate volume.

**NOTE**

If you use the CONFIGURE_AREA, CONFIGURE_IOVF, CONFIGURE_RAA or CONFIGURE_SDEP keyword with the ALLOCATE subcommand to dynamically define the area for an INITIALIZE process, you should not allocate the area to a guaranteed space storage class. If you do so, the guaranteed space storage class overrides the configuration specifications.

Initialization function inputs and outputs

The initialization function has one input and one output as shown in Figure 193.

**Figure 193  Initialization Function Inputs and Outputs**

The area data set is the primary source of input and output for the initialization function.

The initialization function generates the Area Initialization Report. For an example of this report, see the PFPINIT member in the REPORTS data set.
When DBRC is active during the execution of the initialization function, and the area is registered with DBRC, Fast Path/EP verifies from DBRC that the area is marked as “Recovery Needed.” It also obtains authorization for exclusive access (EX) from DBRC for the area data set, as shown in Table 23.

### Table 23  DBRC setting for the initialization function

<table>
<thead>
<tr>
<th>Command</th>
<th>Authorization</th>
<th>Recovery Needed flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>INITIALIZE</td>
<td>exclusive access (EX)</td>
<td>ON</td>
</tr>
</tbody>
</table>

If your areas do not reflect the appropriate Recovery Needed condition, their status must be changed before execution of the initialization function. To set the Recovery Needed flag ON, execute the DBRC utility by using the command shown in Figure 194.

**Figure 194  DBRC command for setting the Recovery Needed flag on**

```
CHANGE.DBDS DBD(dbdbname) AREA (areaname1) RECOV
```

Multiple area data sets (MADS) are not supported. The initialization function initializes only the first area data set that is registered in the ADS list (in collating sequence by DD name). All other area data sets are marked unavailable. After initialization processing has completed, use the IBM Online MADS Create utility to initialize the other (unavailable) area data sets.

When initialization processing is completed, the initialization function resets the “Recovery Needed” flag.

---

**NOTE**

The initialization process also creates a REORG record in DBRC, if DBRC detects that this record is required for recovery.
INITIALIZE command keywords and subcommands

Table 24 lists the keywords available for the INITIALIZE command.

<table>
<thead>
<tr>
<th>Function</th>
<th>Command or subcommand</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>selecting the database and areas</td>
<td>INITIALIZE</td>
<td>DBD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IAREA</td>
</tr>
<tr>
<td>allocating the area data sets</td>
<td>INITIALIZE</td>
<td>INPUT DSN MASK</td>
</tr>
<tr>
<td>defining the area data sets</td>
<td>ALLOCATE</td>
<td>all associated keywords</td>
</tr>
<tr>
<td>initializing associated indexes</td>
<td>IX</td>
<td>all associated keywords</td>
</tr>
<tr>
<td>producing an image copy during initialization</td>
<td>IC</td>
<td>all associated keywords</td>
</tr>
</tbody>
</table>

For more information about the following topics, see the Fast Path/EP Series Reference Manual:

- syntax of commands, subcommands, and keywords that are discussed in this book
- diagrams that show the syntax and available parameters and values for Fast Path Offline Suite commands and subcommands

Selecting the database and areas

The DBD keyword identifies the name of the DEDB (DBD name) to be initialized. The DBD keyword is required with the INITIALIZE command.

The IAREA keyword can be used to select specific areas to be initialized. If you omit the IAREA keyword, all areas that are defined in the DEDB are initialized.

Areas can be specified on the IAREA keyword by using any combination of area names, area numbers, or area ranges. The following parameters are available for the IAREA keyword:

- IAREA=ALL (default) or IAREA=* specifies all areas of the DEDB.

- IAREA=areaname specifies one or more areas by using the one-character to eight-character area name for each area specified. Multiple area names must be enclosed in parentheses and separated by commas.
Allocating the area data sets

- **IAREA=areanumber** specifies one or more areas by using the one-character to five-character area number for each area specified. Multiple area numbers must be enclosed in parentheses and separated by commas.

- **IAREA=(RANGE=(startarea,endarea))** specifies a consecutive range of areas using either areaname or areanumber parameters. The area number associated with startarea must be less than the area number associated with endarea.

An asterisk (*) can be used to specify all areas of the DEDB. When the * character is used with the RANGE keyword, it can be used to specify the beginning or ending range for specific areas of the DEDB.

To initialize an entire DEDB, use a command set like the example shown in Figure 195.

**Figure 195  Sample control statement for initializing all areas**

```
INITIALIZE DBD=dbdname,IAREA=ALL
```

To initialize specific areas, use a command set like the example shown in Figure 196.

**Figure 196  Sample control statement for initializing specific areas**

```
INITIALIZE
  DBD=dbdname,IAREA=(AREANAM1,AREANAM3,RANGE=(5,8))
```

This control statement uses a combination of area names and area numbers to request that the specified areas are to be initialized. The areas named AREANAM1 and AREANAM3 will be initialized. Also, all consecutive areas from area number 5 to area number 8 (area5, area6, area7 and area8) will be initialized.

**Allocating the area data sets**

You can allocate the area for the INITIALIZE command by using traditional DD statements or by using dynamic allocation. When using dynamic allocation, you can also specify certain processing criteria and allocation characteristics for the area.

**Using DD statements**

The areaname DD statement identifies the area data set to be created. If the areaname DD statement is omitted from the JCL, Fast Path Reorg/EP tries to dynamically allocate it.
The area name that is specified on the areaname DD statement must be the same as the area name that is specified in the ACB library. Fast Path Reorg/EP verifies this match by using the IMSACB DD statement.

### Using dynamic allocation to name the area data set

If you are using dynamic allocation to define the area, do not include the areaname DD statement. Fast Path Reorg/EP tries to obtain the data set name for allocation in the following order:

1. If the INPUT_DSN_MASK keyword is specified, it is used to generate the data set name.
2. If DBRC is active and the area is registered, the registered area data set name is obtained from DBRC.
3. The STEPLIB is searched for the DFSMDA member that contains the data set name for this area.

If DBRC is active, and the area is registered with DBRC, the allocated data set name must match the registered data set name, regardless of how it is allocated.

### Using the ALLOCATE subcommand to customize dynamically defined areas

If you elect to dynamically define the areas for the INITIALIZE command, you can specify unique allocation characteristics for the areas by using the ALLOCATE subcommand and its related keywords. The ALLOCATE subcommand can be used as a replacement for DELETE, DEFINE, and other statements that are normally specified by executing the Access Method Services IDCAMS program.

**NOTE**

If the IDCAMS program is already defined in your job input, it is not necessary to remove it. However, any defining statements that are specified under IDCAMS for the area are superseded by similar parameters that are specified on the ALLOCATE subcommand.

By applying the functionality available with the ALLOCATE subcommand, you can accomplish the following tasks and performance improvements:

- automatically adjust the VSAM cluster definition to match an area’s size characteristics
Using the ALLOCATE subcommand to customize dynamically defined areas

- help to simplify JCL streams and job restarts by combining VSAM cluster definition (traditionally defined by IDCAMS) and area processing into a single step
- improve the area’s usage of fragmented DASD space and irregular sized extents
- enhance system I/O performance by promoting parallel processing against the area by easily separating the RAA, IOVF, and SDEP portions of an area across multiple volumes

The ALLOCATE subcommand lets you perform the following tasks:

- delete/define, reuse, or rename the VSAM cluster that is used for the areas
- pass a sequential or partitioned data set to Access Method Services to be used as input for allocation of the new VSAM cluster
- specify the number of volumes and allocation units for the areas
- specify the number of volumes and allocation units for the RAA, IOVF, and SDEP portions of the areas
- pass optional parameters to Access Method Services for the VSAM cluster definition
- specify the SMS storage class, management class, and data class for the areas
- specify space requirements for the areas
- specify volume serial identifiers for the areas

These functions are defined by specifying combinations of keywords that are specified on the ALLOCATE subcommand.

**Identifying the areas for ALLOCATE processing**

The OAREA keyword is specified on the ALLOCATE subcommand to identify the areas for which you want to customize allocation. You can specify one or more areas on a single ALLOCATE subcommand, or specify separate ALLOCATE subcommands for any (or all) of the areas that are identified on the INITIALIZE command’s IAREA keyword.

Areas can be specified on the OAREA keyword under the ALLOCATE subcommand by using any combination of area names, area numbers, or area ranges. Available parameters for the OAREA keyword under the ALLOCATE subcommand are the same as those for the IAREA keyword as specified under the INITIALIZE command. For more information, see “Selecting the database and areas” on page 268.
Specifying the area to be used

The ACTUATE keyword requests optional VSAM cluster processing. The following parameters are available:

- ACTUATE=DELETE causes the current VSAM cluster to be deleted before the new VSAM cluster is allocated.
- ACTUATE=REUSE causes the current VSAM cluster to be allocated and overwritten without allocating a new VSAM cluster.
- ACTUATE=RENAME causes the current VSAM cluster name to be altered according to the data set name mask that you specify before allocating the new VSAM cluster.
- ACTUATE=IDCAMS causes a sequential or partitioned data set to be passed to Access Method Services as input for the allocation of the new VSAM cluster. Due to the design of this feature, each area should have a separate IDCAMS definition.

If you omit the ACTUATE keyword, it is assumed that the area does not already exist. A new area will be allocated.

You can also specify the EROPT parameter on the ACTUATE keyword to specify whether ALLOCATE subcommand processing should continue if a processing error is encountered. Specify EROPT=IGNORE to continue processing. Specify EROPT=ABORT to terminate processing.

Allocating the area and specific portions of the area

Four keywords are available on the ALLOCATE subcommand that control the following allocation characteristics for the specified area or portion of the area:

- size of each extent for the area or portion of the area
- number of extents (volumes) that the area or portion of the area will span

By default, the entire area or any portions of an area (RAA, IOVF or SDEP) that are not explicitly specified with extent size or volume count parameters will be contained on one volume.

The size of the extents are controlled by specifying one or more values on the keywords:

- The CONFIGURE_AREA keyword is used to specify the size of the extent or extents for the entire area data set.
- The CONFIGURE_RAA keyword specifies the size of the extent or extents for the root addressable portion of the area data set.
The CONFIGURE_IOVF keyword specifies the size of the extent or extents for the independent overflow portion of the area data set.

The CONFIGURE_SDEP keyword specifies the size of the extent or extents for the sequential dependent portion of the area data set.

The values that are specified on the CONFIGURE_AREA keyword or any combination of the CONFIGURE_RAA, CONFIGURE_IOVF, and CONFIGURE_SDEP keywords represent the amount of track or cylinder space to be allocated to the extent. Selection of tracks or cylinders for the extents is requested by specifying the SPACE keyword parameter on the ALLOCATE subcommand or is determined by IDCAMS.

To define multiple extents with the same amount of space, you can specify a value on the VOLCNT parameter with an extent size value. The value that is specified on the VOLCNT parameter indicates the number of extents (volumes) that the area or portion of the area will span. Each extent is sized according to the specified extent size.

You can specify the IDCAMS_OPTION keyword on the ALLOCATE subcommand to pass optional parameters to Access Method Services for the VSAM cluster definition. Numerous parameters are available for use with this keyword to control the values that are used by AMS for the VSAM cluster definition. For more information, see the IBM Access Method Services Reference Manual.

**Specifying space requirements, class, and volume identifier for the area**

Other keywords are available that let you specify the SMS class, space requirements, and volume serial identifier for the area. These keywords function in the same manner as on other Fast Path/EP subcommands. These include the AVGREC, DATACLAS, MGMTCLAS, SPACE, STORCLAS and VOLSER keywords.

**ALLOCATE subcommand sample scenarios**

For scenarios that show how the ALLOCATE subcommand can be used with key related keywords to define allocation parameters for areas that are output by INITIALIZE, CHANGE, and RELOAD commands, see Appendix C, “Sample command scenarios.”
Initializing associated indexes

If the DEDB being initialized has registered indexes, you might also want to request that any or all of these indexes be initialized simultaneously with the initialization function.

Specify the IX subcommand and associated keywords for the index to be initialized. For details about the IX subcommand, see the Fast Path Indexer/EP User Guide.

The example in Figure 197 requests that all indexes which are registered to the DEDB be initialized.

Figure 197  Requesting index initialization

<table>
<thead>
<tr>
<th>INITIALIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBD=dbdname, IAREA=ALL</td>
</tr>
<tr>
<td>IX</td>
</tr>
</tbody>
</table>

Producing an image copy during initialization

You can request that one or more output image copy data sets be produced during initialization. Specify the IC subcommand for each image copy to be produced. The example in Figure 198 requests the creation of a single output image copy data set for the area being initialized. For details about the IC subcommand, see

Figure 198  Requesting an image copy during analysis

<table>
<thead>
<tr>
<th>INITIALIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBD=dbdname, IAREA=areaname</td>
</tr>
<tr>
<td>IC DSNAME='dataset-name-mask',UNIT=TAPE,DISP=(NEW,CATLG)</td>
</tr>
</tbody>
</table>

Enhancing performance

Using the initialization function is not necessary when using the CHANGE or RELOAD commands. If the VSAM cluster is empty during execution of the CHANGE or RELOAD command, the initialization function is performed automatically. If the VSAM cluster is not empty, these functions assume that any existing data is to be retained.

For information about CHANGE command performance enhancements, see “Enhancing performance” on page 163. For information about RELOAD command performance enhancements, see “Enhancing performance” on page 231.
Sample INITIALIZE command scenarios

For scenarios that show how to use the INITIALIZE command with related keywords and subcommands, see Appendix C, “Sample command scenarios.”
This chapter provides information about the capabilities, setup, and use of the offline Extend function that is provided by Fast Path Reorg/EP. This function lets you increase the size of IOVF and SDEP storage portions of a DEDB while they are offline to IMS.

This chapter discusses the following topics:

- **Extend function overview**
- **User-specified control of the extend**
- **Automatic pointer validation during the extend**
- **Extend function restrictions**
- **DEBD maintenance requirements**
- **Extend function inputs and outputs**
- **Offline extend considerations**
- **Control statement**
- **Performance considerations**
- **DBRC considerations**
- **Shutting down the Extend function**
- **MADS considerations**
- **EXTEND command keywords and subcommands**
- **Selecting the database and areas**
- **Allocating the area data set**
- **Specifying the execution mode**
- **Extending IOVF and SDEP storage portions**
  - Extending IOVF
  - Extending SDEP
  - Extending IOVF and SDEP by using cylinders
- **Analyzing the DEDB during an extend**
- **Making an image copy during an extend**
- **Sample offline EXTEND command scenarios**
Extend function overview

The patented Extend function of Fast Path Online Reorg/EP offers a high-performance alternative for extending space in an area data set. It provides a quick and easy method for you to increase space without performing a DBD change or a database reorganization.

If an out-of-space condition occurs in either the IOVF or SDEP portion of a DEDB area, you can use the Extend function to increase the size of the IOVF and SDEP portions without performing an online or an offline reorganization. The ability to use this function to extend the IOVF and SDEP portions of the area can save time and resources when compared to executing the traditional unload and reload processes to accommodate the space extend.

The Extend function, which is executed by the EXTEND command, is available for both offline and online processing.

User-specified control of the extend

User-specified keywords let you control the DEDB extend process manually. You can extend all areas or selected areas of a DEDB.

Automatic pointer validation during the extend

If you have a license for the Fast Path Analyzer/EP product, the product performs quick (checksum) pointer validation by default when you execute the Extend function.

Extend function restrictions

The following restrictions apply to the offline Extend function:

- Multiple area data sets (MADS) are not supported for the offline Extend function.
- If you extend IOVF when SDEPs are present in the area, a reorganization of all UOWs will occur.
DEBD maintenance requirements

The following maintenance requirements apply to the offline Extend function when you increase the size of IOVF and SDEP storage portions of a DEDB:

- After extending IOVF, you must update the DBD and ACB with the increased ROOT size when the extend for IOVF has been committed because the ACB is not updated with the increased ROOT parameter.

  If the ACB is not updated and the area is recovered, reloaded, or reinitialized to a smaller-sized IOVF, IMS will not bring the area data set online.

- After extending SDEP, you might need to change your area data set VSAM cluster definition to allow for the extended size of the SDEP portion the next time that you perform maintenance on the DEBD.

Extend function inputs and outputs

The Extend function will dynamically increase IOVF or SDEP space in the existing area data set. The same area data set is used as input to (and output from) the Extend function.

Possible inputs and outputs for an offline Extend function are shown in Figure 199.

Figure 199 Extend function inputs and outputs
The Extend function always issues a summary report to the report log when extension activities are complete. For an example of the Extend Report, see the PFPXTND member in the REPORTS data set.

**Offline extend considerations**

Performing an offline extend requires that the DEDB be taken offline from IMS.

**Control statement**

The PFPSYSIN control statements include the command set necessary to run the offline Extend function. A sample control statement for an offline extend of IOVF is shown in Figure 200. The extend will be limited to only the area that is specified with the IAREA keyword. Dynamic allocation of the area data sets and the ACB library is assumed.

**Figure 200 Sample control statement for an offline extend of IOVF**

```plaintext
//PFP EXEC PGM=PFPMAIN,REGION=0M
//STEPLIB DD DSN=BMC.PFP.LOAD,DISP=SHR
// DD DSN=IMS.RESLIB,DISP=SHR
//PFPSYSIN DD *

EXTEND DBD=dbdname,IAREA=areaname,
   EXTEND_IOVF=(UNITS_OF_WORK,200)
/*
```

**Performance considerations**

The offline Extend function has certain performance considerations.

**Reorganization during the extend**

If you extend IOVF when SDEPs are present in the area, a reorganization of all UOWs will occur.
Analyzing the area during the extend

If you have a license for the Fast Path Analyzer/EP product, the product performs quick (checksum) pointer validation by default during the Extend function. The I/O that is required to read the UOW and IOVF control intervals is shared between the two functions.

For more information, see “Analyzing the DEDB during an extend” on page 290. For example, you can control the level of pointer validation or turn pointer validation off, depending on your performance considerations.

Creating an output image copy during an extend

If you create one or more output image copies while performing an extend, the I/O that is required to read the UOW control intervals is shared between the two functions.

Unlike the extend process, the image copy process requires IOVF control intervals to be processed sequentially. Consequently, the I/O that is required to read the IOVF control intervals for the image copy is not shared. The I/O that is required for processing of the SDEP control intervals is shared between image copy and analysis functions.

When the process is completed, the Extend function notifies DBRC that an image copy is required for each area by setting the “Image Copy Needed” flag. You can use the IC subcommand to request that an image copy be created for each area during the extend. DBRC is informed (NOTIFY.IC) when this image copy is created successfully, which resets the “Image Copy Needed” flag.

If you do not take an image copy during the extend process, a message is issued to indicate that an image copy must be taken before each area can be updated online.

NOTE

BMC recommends that a valid image copy of the input area exists before you execute an offline extend. You should take an image copy of the output area during or after executing the EXTEND command.

Message suppression

For each anomaly that is encountered in an area, the Extend function generates a message with a specific suffix (severity) level. When the number of places that a particular condition exists is large, a large number of messages is produced. You can reduce the number of repetitious messages produced by using the MESSAGE_SUPPRESSION keyword to specify the maximum number of times that
an informational, warning, error or critical message is to be produced. For the extend process, this keyword functions in the same manner as it does for the Fast Path/EP analysis process. For more information, see “Suppressing repetitious messages” on page 111.

**DBRC considerations**

When DBRC is active during the execution of the offline Extend function, and the area is registered with DBRC, an exclusive access (EX) authorization level is requested for the area data set. The data set name for the area data set must match the name that is registered with DBRC.

**Shutting down the Extend function**

When you are running an offline extend, you can terminate the Extend function before completion. Use the operator interface to enter the SHUTDOWN command. Database integrity cannot be guaranteed if termination of the Extend function occurs any other way. Normal backup procedures should be followed before executing the Extend function to provide recovery capability if it is required.

If the SHUTDOWN command is issued before the extend completes, the area extension update will not be committed.

**MADS considerations**

Multiple area data sets (MADS) are *not* supported for offline extends. The offline Extend function searches the area data set (ADS) list that is registered for each area (in collating sequence by ADDN name). The product selects the *first* ADS that is marked as available for use *and* that has no error queue elements (EQEs). If an ADS is found that meets both of these criteria, it is the *only* ADS that will be extended. All other area data sets are marked as *unavailable*. After the extend completes, use the IBM Online MADS Create utility to resynchronize the other (unavailable) area data sets.

If no area data set is marked as available, or if all available area data sets contain one or more EQEs, an extend is not performed for that area.
EXTEND command keywords and subcommands

Table 25 lists the keywords and subcommands that are available for the EXTEND command.

Table 25  EXTEND command keywords and subcommands

<table>
<thead>
<tr>
<th>Function</th>
<th>Command or subcommand</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>selecting the database and area</td>
<td>EXTEND</td>
<td>DBD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IAREA</td>
</tr>
<tr>
<td>allocating the area data set</td>
<td>EXTEND</td>
<td>INPUT_DSN_MASK</td>
</tr>
<tr>
<td>specifying the execution mode</td>
<td>EXTEND</td>
<td>TYPE_RUN</td>
</tr>
<tr>
<td>extending IOVF and SDEP storage</td>
<td>EXTEND</td>
<td>EXTEND_IOVF</td>
</tr>
<tr>
<td>portions</td>
<td></td>
<td>EXTEND_SDEP</td>
</tr>
<tr>
<td>analyzing the DEDB while extending</td>
<td>EXTEND</td>
<td>POINTER_VALIDATION</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RAP_VALIDATION</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDEP_VALIDATION</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LARGEST_DATABASE_RECORDS</td>
</tr>
<tr>
<td>suppressive repetitious messages</td>
<td>EXTEND</td>
<td>MESSAGE_SUPPRESSION</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ORPHANED_SDEP_MSG</td>
</tr>
<tr>
<td>making an image copy of a</td>
<td>IC</td>
<td>all associated keywords</td>
</tr>
<tr>
<td>DEDB while extending</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For more information about the following topics, see the *Fast Path/EP Series Reference Manual*:

- syntax of commands, subcommands, and keywords that are discussed in this book
- diagrams that show the syntax and available parameters and values for Fast Path Offline Suite commands and subcommands

Selecting the database and areas

The DBD keyword identifies the name of the DEDB (DBD name) to be extended. The DBD keyword is required for offline extend processing.
The IAREA keyword can be used to select specific areas to be extended. If you omit the IAREA keyword, all areas defined in the DEDB are extended.

Areas can be specified on the IAREA keyword by using any combination of area names, area numbers, or area ranges. The following parameters are available for the IAREA keyword:

- **IAREA=ALL** (default) or **IAREA=*** specifies all areas of the DEDB.
- **IAREA=areaname** specifies one or more areas by using the one-character to eight-character area name for each area specified. Multiple area names must be enclosed in parentheses and separated by commas.
- **IAREA=areanumber** specifies one or more areas by using the one-character to five-character area number for each area specified. Multiple area numbers must be enclosed in parentheses and separated by commas.
- **IAREA=(RANGE=(startarea,endarea))** specifies a consecutive range of areas using either areaname or areanumber parameters. The area number associated with startarea must be less than the area number associated with endarea.

An asterisk (*) can be used to specify all areas of the DEDB. When the * character is used with the RANGE keyword, it can be used to specify the beginning or ending range for specific areas of the DEDB.

To extend IOVF for an entire DEDB, use a command set like the example shown in Figure 201.

**Figure 201  Sample control statement for extending IOVF for all areas**

```plaintext
EXTEND DBD=dbdname, IAREA=ALL, EXTEND_IOVF=(UNITS_OF_WORK,100)
```

To extend SDEPs for specific areas, use a command set like the example shown in Figure 202.

**Figure 202  Sample control statement for extending SDEPs for specific areas**

```plaintext
EXTEND DBD=dbdname, IAREA=(AREANAM1,AREANAM3,RANGE=(5,8)), EXTEND_SDEP=(CONTROL_INTERVALS,100)
```

This control statement uses a combination of area names and area numbers to request that the specified areas are to be extended. The SDEP portion in areas named AREANAM1 and AREANAM3 will be extended. Also, all consecutive areas from area number 5 to area number 8 (area5, area6, area7 and area8) will be extended.
Allocating the area data set

For an offline extend, the area data set can be supplied in the JCL or can be accessed by using dynamic allocation. The areaname DD statement identifies the area data set to be extended. If the areaname DD statement is omitted from the JCL, Fast Path Reorg/EP tries to dynamically allocate it.

The IMSACB DD statement identifies the ACB library containing the database definition that describes the area that is referenced by the areaname DD statement.

If you are using dynamic allocation, do not include the areaname DD statement. Fast Path Reorg/EP tries to obtain the data set name for allocation in the following order:

1. If the INPUT_DSN_MASK keyword is specified, it is used to generate the data set name.
2. If DBRC is active and the area is registered, the registered area data set name is obtained from DBRC.
3. The STEPLIB is searched for the DFSMDA member that contains the data set name for this area.

If DBRC is active and the area is registered with DBRC, the allocated data set name must match the registered data set name regardless of how it is allocated.

Specifying the execution mode

When using the EXTEND command, you can use the TYPE_RUN keyword to specify the execution mode for an area extension. The following values are available:

- EXECUTE (default) performs an area extension.
- SIMULATE imitates (models) an area extension without actually executing the extension.

For more information about the TYPE_RUN keyword, see the Fast Path/EP Series Reference Manual.
Extending IOVF and SDEP storage portions

If an out-of-space condition occurs in IOVF or SDEP, you can use the Fast Path Reorg/EP patented Extend feature to extend the IOVF or SDEP portions of a DEDB. The ability to extend the IOVF or SDEP portions of the area can save time and resources in comparison to using offline CHANGE or UNLOAD/RELOAD commands.

Depending on how many IOVF or SDEP portions you request, the number of control intervals (CIs) that are required to accommodate your request is rounded up to the next control area (CA) boundary. If additional CIs must be added as a result of the rounding, the additional CIs are added to the SDEP portion of the database.

IOVF can be extended online when there are SDEPs defined. However, the minimum number of blocks that are being added must be as large as the number of SDEP blocks that are currently in use. The number of SDEP blocks that are in use can be reduced by executing the SDEP Scan/Delete utilities before executing the Extend function.

If you are extending IOVF, and the SDEP portion of the area is in a wrapped condition, the SDEP blocks will be rearranged to unwrap the SDEP portion.

If you are extending SDEP only, and the SDEP portion of the area is in a wrapped condition, the SDEP blocks will not be unwrapped. The existing SDEP blocks are rearranged so that the newly created SDEP blocks fall between the SDEP Logical End (LE) and SDEP Logical Begin (LB) points. As a result, the new blocks are immediately available for use; there is no need to run the SDEP Scan/Delete utilities. This action occurs only if the number of new SDEP blocks exceeds the number of used SDEP blocks between the LB point and the original end of file.

The following sections present step-by-step procedures for using the EXTEND command to extend IOVF or SDEP.

Extending IOVF

The IOVF portion can be extended by specifying units of work, control intervals, or cylinders. The EXTEND_IOVF keyword allows the EXTEND command to extend the IOVF portion.
To extend the IOVF

1 Ensure that adequate space is present on the primary volume of the target area data set to accommodate the request before attempting to increase the IOVF portion during the Extend function. If adequate space is not present, allocate additional volumes to the area data set by using IDCAMS ALTER ADDVOLUME.

2 Run the Extend function by using the EXTEND_IOVF keyword to specify the amount of space to add.

3 When the Extend function is complete, message BMC111193I is issued to the processing log, showing the new ROOT parameter for the DBD source. Use the values that are indicated in message BMC111193I to update the DBD definition to include the IOVF extension. Message BMC110000I is also issued, stating that the extension has been committed.

Sample control statements for extending IOVF

Sample control statements for extending IOVF are shown in Figure 203.

Sample output

The following example shows messages BMC111193I and BMC110000I that are generated when adding 50 units of work to the IOVF portion:

BMC111193I dbdname, areaname: ROOT parameter changed from (200,100) to (250,150)

BMC110000I dbdname, areaname: AREA extension committed date, time

When the database has been started for IMS, the following message, DFS3703I, is issued by IMS in the control region the first time that the area is opened:

DFS3703I Size of independent overflow has been increased for area areaname
Subsequent maintenance on the DEDB

When the extend for IOVF has been committed, the ACB is *not* updated with the increased ROOT parameter. You must update the DBD and the ACB with the increased ROOT size.

If the ACB is not updated and the area is ever recovered, reloaded, or reinitialized to a smaller-sized IOVF, IMS will not bring the area data set online.

*To ensure that the extended IOVF is applied to the DBD definition and that the ACB definition is revised*

1. Update the ROOT parameter of the AREA statement in the DBD source with the values that you recorded from message BMC111193I.
2. Run DBDGEN.
3. Run ACBGEN.

*NOTE*
You might need to change your area data set VSAM cluster definition to reflect the extended size.

Extending SDEP

The SDEP storage portion can be extended by specifying control intervals, units of work, or cylinders. To extend the SDEP storage area, use the EXTEND_SDEP keyword with the EXTEND command to specify the amount of space to be added.

*To extend SDEP storage*

1. Ensure that adequate space is present on the primary volume of the target area data set to accommodate the request before trying to increase the SDEP portion during the Extend function. If adequate space is not present, allocate additional volumes to the area data set by using IDCAMS ALTER ADDVOLUME.
2. Before running the Extend function with the EXTEND_SDEP keyword, BMC recommends that you execute the SDEP Scan/Delete utilities.
3 Run the Extend function with the EXTEND_SDEP keyword to specify the amount of space to add.

Sample control statement for extending SDEP

A sample control statement for extending SDEP is shown in Figure 204.

Figure 204  Sample control statements for extending SDEP

```
EXTEND DBD=dbdname, IAREA=areaname,
     EXTEND_SDEP=(CONTROL_INTERVALS,100)
```

Sample output

The following example shows message BMC110000I that is generated when adding control intervals to the SDEP portion:

BMC110000I dbdname, areaname: AREA extension committed date, time

Subsequent maintenance on the DEDB

The next time that you perform maintenance on the DEDB, you might need to change your area data set VSAM cluster definition to allow for the extended size of the SDEP portion.

Extending IOVF and SDEP by using cylinders

In addition to using control intervals and units of work, you can also use cylinders to extend IOVF and SDEP storage. A sample control statement for extending IOVF and SDEP by using cylinders is shown in Figure 205.

Figure 205  Sample control statements for extending IOVF by 100 cylinders and SDEP by 50 cylinders

```
EXTEND DBD=dbdname, IAREA=areaname,
     EXTEND_IOVF=(CYLINDERS,100), EXTEND_SDEP=(CYLINDERS,50)
```
Analyzing the DEDB during an extend

If your site has a license for the Fast Path Analyzer/EP product, pointer validation will occur automatically (by default) when you execute an offline extend.

This automatic process applies the default value of QUICK for the POINTER_VALIDATION keyword. The analysis function performs a checksum validation of pointers for each area of the database that is specified on the EXTEND command. Automatic pointer validation provides assurance of the area’s pointer integrity, while providing statistics that show how the extend process affected space usage.

NOTE
You can override automatic quick pointer validation and request more detailed validation by specifying POINTER_VALIDATION=FULL on the EXTEND or GLOBAL command. Direct pointers are validated by using the cross-reference technique. Although not recommended, you can also specify another valid value for the POINTER_VALIDATION keyword (such as POINTER_VALIDATION=NONE) to completely disable the analysis function.

If an SDEP segment is defined for the database and the number of UOWs for IOVF control intervals is being extended (using the EXTEND_IOVF keyword), SDEP pointers are also validated (by default) by using the same technique as specified for direct pointers. You can manually specify any valid value for the SDEP_VALIDATION keyword to control how SDEP pointers are to be validated.

You can also specify additional functions that are associated with the analysis function:

- Use the RAP_VALIDATION keyword to control how RAPs are processed.
- Use the LARGEST_DATABASE_RECORDS keyword to specify the number of largest database records to be tracked by the analysis process.
- Use the REPORT and THRESHOLD subcommands to control the generation of analysis reports and exception testing.

For more information about these keywords and subcommands, see Chapter 4, “Analyzing a DEDB.”

Because no values are specified for the POINTER_VALIDATION and SDEP_VALIDATION keywords, the example in Figure 206 requests that pointer validation be performed during an extension of the area by using the QUICK (checksum) technique and that no validation of SDEP pointers is performed. Full validation for RAP pointers is requested explicitly. The example produces only the Free Space Analysis report and does not check threshold conditions.
You can request that one or more output image copy data sets be produced during an extend. Specify the IC subcommand for each image copy to be produced. For details about the IC subcommand, see Chapter 10, “Making an image copy.”

The example in Figure 207 requests the creation of a single output image copy data set for the area being extended.

For scenarios that show how to use the EXTEND command with key related keywords and subcommands, see Appendix C, “Sample command scenarios.”
Making an image copy

This chapter provides information on the offline image copy function. This chapter discusses the following topics:

Offline image copy function overview ............................................................ 293
Image copy function inputs and outputs ......................................................... 294
IC subcommand keywords and supported commands .................................... 294
Selecting the database and areas ................................................................. 295
  Making an image copy during analysis ....................................................... 295
  Making an image copy during reorganization .............................................. 296
  Making an image copy of a changed DEDB ............................................... 296
  Making an image copy during unload ......................................................... 297
  Making an image copy during reload ......................................................... 297
Allocating the image copy data set .............................................................. 298
  Using JCL to allocate the image copy data set ......................................... 298
  Using dynamic allocation for the image copy data set ............................... 299
Stacking image copy data sets ................................................................. 299
Generating multiple output image copies .................................................. 301
Recording the image copy within DBRC ..................................................... 301
Compressing the output image copy ......................................................... 302
Sample IC subcommand scenarios ............................................................. 303

Offline image copy function overview

Many Fast Path Offline Suite product functions provide the facility to create one or more image copies of the DEDB simultaneously with the execution of the function. These functions include offline analysis, change, unload, reload, and reorganization. With each function, an image copy is requested by specifying the IC subcommand.
Image copy function inputs and outputs

The image copy function has one input and one or more outputs as shown in Figure 208.

Figure 208  Image copy function inputs and outputs

IC subcommand keywords and supported commands

Fast Path/EP primary commands that are available to the IC subcommand and keywords available for the IC subcommand are shown in Table 26.
Selecting the database and areas

Chapter 10 Making an image copy 295

For more information about the following topics, see the Fast Path/EP Series Reference Manual:

- syntax of commands, subcommands, and keywords that are discussed in this book
- diagrams that show the syntax and available parameters and values for Fast Path Offline Suite commands and subcommands

### Selecting the database and areas

The database and areas to be image–copied are specified by a Fast Path/EP primary command. Each primary command is discussed in this section.

On the command, the DBD keyword identifies the name of the DEDB to be image–copied when you specify the IC subcommand. The IAREA keyword can be used to select specific areas to be image–copied. You can specify one or more area names. If you omit the IAREA keyword, *all* areas that are defined in the DEDB are image–copied.

### Making an image copy during analysis

You can request that one or more image copy data sets be created for an area during the analysis function. Use the IC subcommand to request creation of an image copy.

---

**Table 26 Command keywords for the IC subcommand**

<table>
<thead>
<tr>
<th>Function</th>
<th>Command or subcommand</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>selecting the database and areas to be image copied</td>
<td>ANALYZE CHANGE EXTEND INITIALIZE RELOAD REORGANIZE UNLOAD</td>
<td>DBD IAREA</td>
</tr>
<tr>
<td>allocating the image copy data set</td>
<td>IC</td>
<td>DDNAME DSNAME</td>
</tr>
<tr>
<td>stacking image copy data sets</td>
<td>IC</td>
<td>STACK_NAME</td>
</tr>
<tr>
<td>generating multiple output image copies</td>
<td>IC</td>
<td>all associated keywords</td>
</tr>
<tr>
<td>recording the image copy with DBRC</td>
<td>IC</td>
<td>NOTIFY</td>
</tr>
<tr>
<td>compressing the output image copy</td>
<td>IC</td>
<td>COMPRESSION</td>
</tr>
</tbody>
</table>

---

For more information about the following topics, see the Fast Path/EP Series Reference Manual:
Making an image copy during reorganization

The example in Figure 209 requests the creation of two output image copy data sets for each area within the database as the areas are being analyzed.

**Figure 209  Specifying the image copy function with the ANALYZE command**

```
ANALYZE DBD=dbdname,AREA=ALL
  IC DSN='dataset-name1-mask',
       DISP=(NEW,CATLG),UNIT=TAPE
  IC DSN='dataset-name2-mask',
       DISP=(NEW,CATLG),UNIT=TAPE
```

For more information about analyzing areas, see Chapter 4, “Analyzing a DEDB.”

Making an image copy during reorganization

You can request that one or more image copy data sets be created for an area during the reorganization function. Use the IC subcommand to request creation of an image copy.

The example in Figure 210 requests the creation of two output image copy data sets for each area within the database as the areas are being reorganized.

**Figure 210  Specifying the image copy function with the REORGANIZE command**

```
REORGANIZE DBD=dbdname,AREA=ALL
  IC DSN='dataset-name1-mask',
       DISP=(NEW,CATLG),UNIT=TAPE
  IC DSN='dataset-name2-mask',
       DISP=(NEW,CATLG),UNIT=TAPE
```

For more information about reorganizing areas, see Chapter 3, “Reorganizing a DEDB.”

Making an image copy of a changed DEDB

You can request that one or more image copy data sets be created for each output area that is created during the DEDB change function. Use the IC subcommand to request creation of an image copy.

The example in Figure 211 requests the creation of one output image copy data set for each of two areas (area1 and area2) within the changed database.
Making an image copy during unload

You can request that one or more image copy data sets be created for an area during the unload function. Use the IC subcommand to request creation of an image copy.

The example in Figure 212 requests the creation of two output image copy data sets for each area within the database as the areas are being unloaded.

Figure 212  Specifying the image copy function with the UNLOAD command

<table>
<thead>
<tr>
<th>UNLOAD DBD=dbdname,IAREA=ALL</th>
<th>IC DSNAME='dataset-name1-mask', DISP=(NEW,CATLG),UNIT=TAPE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IC DSNAME='dataset-name2-mask', DISP=(NEW,CATLG),UNIT=TAPE</td>
</tr>
</tbody>
</table>

For more information about the unload function, see Chapter 6, “Unloading and Reloading a DEDB.”

Making an image copy during reload

You can request that one or more image copy data sets be created for each output area that is created during the reload function. Use the IC subcommand to request creation of an image copy.

The example in Figure 213 requests the creation of one output image copy data set for each of two areas (area1 and area2) within the reloaded database.

Figure 213  Specifying the image copy function with the RELOAD command

| RELOAD DBD=dbdname,IAREA=ALL,OAAREA=(area1,area2) | IC DSNAME='dataset-name-mask', DISP=(NEW,CATLG),UNIT=TAPE |

For more information about the change function, see Chapter 5, “Changing DEDB structure in place.”
Allocating the image copy data set

The output image copy data set can be allocated by using DD statements supplied in the JCL or by using dynamic allocation. BMC recommends dynamic allocation because of the flexibility and improved control that it provides.

Using JCL to allocate the image copy data set

Use the DDNAME keyword to indicate that the image copy data set is allocated using JCL DD statements.

The example in Figure 214 shows the request to make an image copy of an area. The output image copy is written by the DD statement named `icddname` that is supplied in the JCL.

**Figure 214  JCL to allocate an image copy data set**

```plaintext
//PFP EXEC PGM=PFPMAIN,REGION=0M
//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
// BMC.ICP.LOAD,DISP=SHR
// DD DISP=SHR,DSN=IMS.RESLIB
//icddname DD UNIT=TAPE,DISP=(NEW,CATLG),
// DSN=your.dataset.name
//PFPSYSIN DD *
ANALYZE DBD=dbdname,IAREA=areaname
IC DDNAME=icddname
/*

**NOTE**

When using JCL to allocate the output image copy, you should ensure that only one area is written to any DD statement. If more than one area is written to the same DD statement, the image copy of the second area overwrites the image copy of the first area.
Using dynamic allocation for the image copy data set

Use the DSNAME keyword to indicate that the image copy data set is to be dynamically allocated and to supply a mask for constructing the data set name. Substitution variables within the mask can be used to generate unique names for each output data set.

Additional keywords such as UNIT, DISP and DATACLAS can be used to control the allocation and disposition of the data set, similar to the corresponding JCL keywords.

The example in Figure 215 requests the creation of a single output image copy data set for each area (using the default IAREA=ALL) within the database. Each image copy data set is written to a tape device and recorded in the system catalog.

Figure 215  Dynamic allocation of an image copy data set

<table>
<thead>
<tr>
<th>REORGANIZE  DBD=dbname</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC DSNAME='dataset-name-mask',</td>
</tr>
<tr>
<td>DISP=(NEW,CATLG),UNIT=TAPE</td>
</tr>
</tbody>
</table>

The data set name that is specified with the DSNAME keyword can refer to a generation data group (GDG). Code the DSNAME keyword to specify the base name for the GDG. You can include the (+1) relative generation number as part of the DSNAME keyword.

The example in Figure 216 uses a DSNAME keyword to refer to a GDG.

Figure 216  Using a generation data set group

<table>
<thead>
<tr>
<th>REORGANIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC DSNAME='dataset-name-mask(+1)',</td>
</tr>
<tr>
<td>DISP=(NEW,CATLG),UNIT=TAPE</td>
</tr>
</tbody>
</table>

If you do not specify the (+1) relative generation number, the product automatically appends the incremented (+1) generation number to the data set name if it is recorded in the system catalog as a GDG base name.

Stacking image copy data sets

Use the STACK_NAME keyword to specify the name of a stack group to which the image copy data set belongs. The same name is specified for each image copy that is to be included in the group. All image copy data sets that are part of the same stack group are written onto the same tape volume (or volumes) as file number 1, 2, 3, and so on.
Figure 217 specifies that an image copy data set be created for each area in the database. The output data sets are all written to the same tape volume.

**Figure 217  Using the STACK_NAME keyword**

```
REORGANIZE DBD=dbdname,IAREA=ALL
   IC STACK_NAME=NAME1,UNIT=TAPE,DISP=(NEW,CATLG),
       DSNAME='dataset-name-mask'
```

Dynamic allocation is always used for stacked data sets. When an IC subcommand containing a unique value for the STACK_NAME keyword is encountered, values that are specified for the UNIT, VOLCNT, VOLSER, EXPDT and RETPD keywords (if any) on the IC subcommand are saved; these values will apply to all image copy data sets that are written to the stack. The UNIT keyword must be specified and must refer to a tape device type. All other dynamic allocation keywords, such as DSNAME, pertain to the individual output image copy data set that is being defined. DISP=NEW is required for stacked data sets; there are no restrictions on the remaining parameters for the DISP keyword.

As the image copy function is executing, only one output image copy data set within a stack group can be open at a time. You must not specify the same stack group on more than one IC subcommand within any primary function command, such as IMAGECOPY.

In Figure 218, two image copy data sets are requested for each area in the database. You cannot request that both copies be written to the same stack. You can, however, specify a different STACK_NAME keyword on each IC subcommand. You can also mix stacked and non-stacked image copy processing within the same primary command.

**Figure 218  Illegal use of the same stack name within a single command**

```
IMAGECOPY DBD=dbdname,IAREA=ALL, ...
   IC STACK_NAME=NAME1, ...
   IC STACK_NAME=NAME1, ...
```

Another consequence of this restriction applies when the product is performing parallel processing of commands in the offline mode. Because only one output image copy data set within a stack group can be open at a time, the serialization of each stack group affects the amount of parallel processing that is performed.

In Figure 219, three separate parallel tasks are created (one for each REORGANIZE command). All of these tasks, however, are also creating image copy output data sets within the same stack group. Consequently, only one reorganization task can be performed at a time; the other two tasks will be waiting for their turn to access the output image copy stack.
Generating multiple output image copies

One output image copy data set is produced for each IC subcommand that is specified. The product imposes no limit on the number of duplicate image copies that can be produced simultaneously.

The example in Figure 220 shows a request to make three image copy data sets for each area within the database. The first and second image copy data sets are written to tape devices and are recorded in the system catalog. The third image copy is written to an existing catalogued data set and overwrites the prior contents of the data set.

Figure 220 Specifying multiple image copy data sets for each area

```
REORGANIZE DBD=dbdname, IAREA=ALL, ...
IC DSNAME='dataset-name1-mask',
        DISP=(NEW,CATLG),UNIT=TAPE
REORGANIZE DBD=dbdname1, IAREA=ALL, ...
IC DSNAME='dataset-name2-mask',
        DISP=(NEW,CATLG),UNIT=TAPE
REORGANIZE DBD=dbdname2, IAREA=ALL, ...
IC DSNAME='dataset-name3-mask',DISP=SHR
```

Recording the image copy within DBRC

If DBRC is active, and if the area being processed is registered within DBRC, the image copy data set or data sets that are produced are recorded using the NOTIFY.IC process. There is no limit to the number of image copy data sets you can create, but no more than two image copy data sets can be recorded within DBRC. You can control which image copy data set or data sets are recorded by using the NOTIFY keyword:

- NOTIFY=YES (the default) indicates that the image copy data set is eligible to be recorded in DBRC unless an error occurs during the creation of the image copy data set.
- NOTIFY=NO indicates that the image copy data set is not eligible to be recorded in DBRC even if no errors occur during the creation of the image copy data set. This option can be useful when producing an image copy that is intended for offsite recovery purposes.
If more than two image copy data sets are created successfully, the first two eligible image copies (in the order that the IC subcommands are specified) are recorded within DBRC.

The example in Figure 221 requests the creation of three output image copies. The first two image copies are eligible to be recorded within DBRC, but the third image copy is not eligible. If an error occurs during creation of the first image copy, it is not recorded; the second image copy is the only image copy recorded, since the third image copy is not eligible because NOTIFY=NO is specified. If NOTIFY=YES was specified on the third image copy, it would have been recorded in DBRC if an error had occurred during creation of the first image copy.

**Figure 221  Recording image copies with DBRC**

```
REORGANIZE DBD=dbdname, IAREA=areaname
IC DDNAME=ddname1, NOTIFY=YES
IC DDNAME=ddname2, NOTIFY=YES
IC DDNAME=ddname3, NOTIFY=NO
```
The example in Figure 222 requests the creation of three output image copies. The first image copy is not compressed. The second image copy data set is compressed by using the FSE algorithm. The third image copy data set is compressed by using the CCC algorithm.

**Figure 222  Compressing the image copy data set**

<table>
<thead>
<tr>
<th>Command</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMAGECOPY</td>
<td>IAREA=areaname</td>
</tr>
<tr>
<td>IC</td>
<td>DDNAME=ddname1,COMPRESSION=NONE</td>
</tr>
<tr>
<td>IC</td>
<td>DDNAME=ddname2,COMPRESSION=FSE</td>
</tr>
<tr>
<td>IC</td>
<td>DDNAME=ddname3,COMPRESSION=CCC</td>
</tr>
</tbody>
</table>

**Sample IC subcommand scenarios**

For scenarios that show how to use the IC subcommand with key Fast Path Offline Suite primary commands, see Appendix C, “Sample command scenarios.”
Chapter 11 Statistics repository facility

This chapter provides information on the use of the statistics repository facility that is provided by the Fast Path Analyzer/EP. This repository is a collection of data sets containing statistics that are gathered during analysis processes. This information can be used for historical and trend analysis. This chapter discusses the following topics:

Repository overview .................................................. 306
    Repository contents ........................................... 306
    Repository inputs and outputs ............................ 307
Allocation rules ................................................... 308
    Allocation rule parameters ............................... 308
    Default allocation rule .................................. 310
Processing the repository ...................................... 310
Creating a repository catalog ................................. 311
Creating and modifying allocation rules .................... 312
    Listing allocation rules ................................. 313
    Adding allocation rules ................................. 314
    Modifying allocation rules ............................ 315
    Deleting allocation rules ............................... 316
Activating repository processing ............................. 317
    Activating the repository by using JCL ................ 317
    Activating the repository by using dynamic allocation of the catalog on OPTIONS command ................... 318
    Activating dynamic allocation of the repository by using DFSMDA member .................................................................................. 318
    Disabling dynamic allocation of the repository .......... 319
    Specifying the repository group ........................... 319
Creating statistics data sets ................................. 320
    Allocating the statistics data set ....................... 320
    Specifying the detail level ............................... 321
    Controlling retention of repository elements ........... 322
Managing statistics catalog entries .......................... 322
    Listing statistics catalog entries ..................... 323
    Adding statistics catalog entries ..................... 324
    Deleting statistics catalog entries .................... 324
Retrieving statistics from the repository .................. 327
    Producing Reports ......................................... 327
    Producing a history file .................................. 328
Repository overview

The repository is an optional facility that provides a mechanism for the storage of the statistics gathered during the analysis function, and for the retrieval of these statistics for subsequent processing. If the repository facility is activated, statistical information gathered by the analysis function will be saved in the repository for future processing. If the repository facility is not activated, statistical information is not retained after the generation of the reports that are requested for the DEDB area.

The repository facility is available to the online analysis process if you have a license for the Fast Path Online Analyzer/EP product. It is available to the offline analysis process if you have a license for Fast Path Online Analyzer/EP or Fast Path Analyzer/EP.

The repository facility enables you to accomplish several things:

- store the statistics gathered by the analysis function
- retrieve the statistics for reporting purposes
- use the statistics to perform historical reporting, trend analysis and forecasting
- store, view, or delete customizations to the severity level or issuance threshold (the number of times it is issued before it is suppressed) for selected product messages

This chapter discusses the use of the repository for statistics storage, retrieval and usage. For information about using the repository for message customization, see Appendix B, “Dynamic message modification.”

Repository contents

The repository consists of a single catalog data set called a repository catalog, and a number of statistics data sets.

The repository catalog is a VSAM keyed sequential data set (KSDS). It contains a set of statistics catalog entries, and a set of allocation rules that control the creation and retention of new statistics data sets.

Each statistics data set is a sequential file that contains the detailed statistical information that is gathered by a single execution of the analysis function for a DEDB or area.
Statistics catalog entries

The analysis function creates a statistics catalog entry in the repository catalog for each DEDB area that it processes. The statistics catalog entry identifies the DEDB area that is analyzed and the date and time that the analysis was performed. The statistics catalog entry also contains some summary statistics that are gathered during analysis and specifies the name of the statistics data set (if any) that contains the more detailed statistical information.

Allocation rules

The allocation rules contain information that is needed to control the allocation of a statistics data set that contains the detailed statistical information which is gathered during the analysis. The rules specify how the data set name is to be determined, details about the type of storage unit to be used, and specific volume serial numbers. If the rules specify that the data set name is to be NULLFILE, no statistics data set is allocated.

Repository inputs and outputs

Figure 223 shows the inputs and outputs for the repository facility.

Figure 223 Repository Processing

---

Figure 223 shows the inputs and outputs for the repository facility.
The bidirectional arrows between the Analyzer products and the repository (the repository catalog and the statistics data sets) indicate that the product writes data to the repository and that it can also retrieve information from the repository to generate report output.

**Allocation rules**

Allocation rules are used by the analysis function to create new repository statistics data sets. An allocation rule is selected by the analysis function to control this process. Selection of the allocation rule to be used is described in “Creating statistics data sets” on page 320.

**Allocation rule parameters**

An allocation rule contains user-supplied information that is based on values which are specified for the keywords discussed in this section.

**DSNAME**

Each allocation rule contains a data set name mask that is used to build the name of the statistics data set. The mask consists of a string of literal characters and substitution parameters. Table 27 lists the substitution parameters that are available for building a statistics data set name:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;DBD</td>
<td>DBD name</td>
</tr>
<tr>
<td>&amp;AREA</td>
<td>area name</td>
</tr>
<tr>
<td>&amp;AREA#</td>
<td>three-digit area number</td>
</tr>
<tr>
<td>&amp;AREA4#</td>
<td>four-digit area number</td>
</tr>
<tr>
<td>&amp;AREA5#</td>
<td>five-digit area number</td>
</tr>
<tr>
<td>&amp;IMSID</td>
<td>repository group</td>
</tr>
<tr>
<td>&amp;DATE</td>
<td>current date (“Dyyddd”)</td>
</tr>
<tr>
<td>&amp;TIME</td>
<td>current time (“Thhmmss”)</td>
</tr>
<tr>
<td>&amp;JOBN</td>
<td>JOB name</td>
</tr>
<tr>
<td>&amp;STEPN</td>
<td>STEP name</td>
</tr>
<tr>
<td>&amp;PROCSN</td>
<td>PROC STEP name</td>
</tr>
</tbody>
</table>
The purpose of the data set name mask and substitution parameters is to generate a unique name for each statistics data set. A good technique for creating unique names is to use &DATE and &TIME substitution parameters as part of the data set name.

**Generation data group**

You can also generate a unique statistics data set name by using generation data groups (GDGs). Use the data set name mask to construct the GDG base name, but do not include relative or absolute generation numbers. The analysis function automatically appends the next generation number to the data set name if it detects that the name is a GDG base name.

**NULLFILE as data set name**

The generated data set name has a special significance when the first eight characters of the generated data set name is NULLFILE. When this data set name is specified, no statistics data set is created. The statistics catalog entry is still added, however, to identify that analysis was performed for the area and to store the summary statistics that are gathered during analysis.

When no statistics data set is created, or if the statistics data set has been deleted or is not available, only the summary statistics that are stored in the statistics catalog entry are available for subsequent processing. The summary statistics are not sufficient for regeneration of all analysis reports, but can be used to generate the following information:

- Area Summary Report
- abbreviated version of the Free Space Analysis Report
- information stored in the History File (For more information, see the *Fast Path/EP Series Reference Manual.*

**UNIT**

Each allocation rule contains a unit name that specifies the device type where the new statistics data set is to be stored.

**EXPDT and RETPD**

Each allocation rule contains an expiration date (EXPDT) or a retention period (RETPD) to be associated with the new statistics data set when it is allocated.
DATACLAS, MGMTCLAS, and STORCLAS

Each allocation rule contains any combination of SMS data class, management class, and storage class.

VOLSER

Each allocation rule contains a volume serial number (VOLSER) to specify the storage volume where the new statistics data set will be placed.

Default allocation rule

When a repository catalog is initialized, a default global allocation rule is inserted. This default rule specifies the DSNAME parameter as NULLFILE, and the UNIT parameter as SYSDA. By default, no statistics data set is created for any DEDB area. However, the statistics catalog entries containing the summary statistics are still created.

You might want to modify the allocation rule (or rules) so that the analysis function will create the more detailed statistics data sets. To set up your own allocation rules, see “Creating and modifying allocation rules” on page 312.

Processing the repository

There are several ways that you can process a repository.

- Create a repository catalog

  A repository catalog is created when you install Fast Path Analyzer/EP or Fast Path Online Analyzer/EP. However, you might want to have more than one repository catalog. If you want to create an additional repository catalog, you must allocate a VSAM KSDS to store it, then initialize it by executing the PFPEPR00 offline utility program using the PROCESS_EPR command.

- Create and modify allocation rules

  You can set up allocation rules using the PROCESS_EPR command and its associated subcommands and keywords available in PFPEPR00. You can also list, modify, or delete allocation rules.
- **Activate repository processing**

  Repository processing is activated when requested during execution of either an online or offline analysis process.

- **Manage the statistics catalog entries**

  You can perform maintenance on the repository as needed using the PROCESS_EPR command and its associated subcommands and keywords available in the PFPEPR00 offline utility program. It also includes adding, listing or deleting statistics catalog entries for statistics data sets.

- **Retrieve statistics from the repository**

  Several methods are available for retrieving information from the repository to generate reports or create a historical file for other analysis mechanisms.

- **Use Fast Path Manager/EP ISPF panels**

  You can use Fast Path Manager/EP ISPF panels as an optional method to view the repository catalog, delete statistics catalog entries from the repository catalog, or request reports.

---

**Creating a repository catalog**

One repository catalog is created automatically at the time you install any Fast Path Offline Suite product. However, you can allocate and initialize any number of additional repository catalogs. You might prefer, for example, to use one repository catalog for your test environment and a different repository catalog for your production environment.

Creating an additional repository catalog is a two-step process:

1. Execute the IBM IDCAMS utility to create a VSAM KSDS.

2. Initialize the new VSAM KSDS by executing the PFPEPR00 program. The JCL to create a repository catalog is built during product installation. Sample JCL can also be found in the Fast Path/EP sample library in member name #INITEPR. For more information, see “Create and initialize a repository catalog” on page 422.
Creating and modifying allocation rules

The PFPEPR00 utility program offers four subcommands that can be specified on the PROCESS_EPR command to manage allocation rules: LIST, ADD, MODIFY and DELETE. When combined with a rule-type object identifier, these subcommands let you manage allocation rules. An allocation rule can be defined at the global, group, DBD, or area level. Table 28 lists allocation management commands and subcommands that are available.

Table 28  Allocation commands and subcommands available with PFPEPR00

<table>
<thead>
<tr>
<th>Function</th>
<th>Command, subcommand, or object</th>
<th>Available keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>selecting repository catalog for processing</td>
<td>PROCESS_EPR</td>
<td>REPOSITORY_DSNAME (required)</td>
</tr>
<tr>
<td>listing allocation rules</td>
<td>LIST type_ALLOCATION (where type=GLOBAL, GROUP, DBD or AREA) or LIST ALLOCATION (generic type)</td>
<td>SELECT_AREA SELECT_DBD SELECT_GROUP</td>
</tr>
<tr>
<td>adding allocation rules</td>
<td>ADD type_ALLOCATION (where type=GLOBAL, GROUP, DBD or AREA)</td>
<td>AREA_KEY DATACLAS DETAIL_LEVEL DBD_KEY DSNAME (required) EXPDT GROUP_KEY MGMTCLAS RETPD STORCLAS UNIT (required) VOLSER</td>
</tr>
<tr>
<td>modifying allocation rules</td>
<td>MODIFY type_ALLOCATION (where type=GLOBAL, GROUP, DBD or AREA) or MODIFY ALLOCATION (generic type)</td>
<td>DATACLAS DETAIL_LEVEL DSNAME EXPDT MGMTCLAS RETPD STORCLAS UNIT VOLSER SELECT_AREA SELECT_DBD SELECT_GROUP</td>
</tr>
<tr>
<td>deleting allocation rules</td>
<td>DELETE type_ALLOCATION (where type=GLOBAL, GROUP, DBD or AREA) or DELETE ALLOCATION (generic type)</td>
<td>SELECT_AREA SELECT_DBD SELECT_GROUP</td>
</tr>
</tbody>
</table>
Listing allocation rules

The LIST subcommand is useful when you want to list allocation rules contained in a repository catalog. Specify the LIST subcommand in the following format:

```
LIST_type_ALLOCATION
```

The variable `type` refers to the level at which you want to list the allocation rule (Table 29).

**Table 29  Type parameter values**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Allocation rule listed</th>
<th>Keyword requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLOBAL</td>
<td>global level</td>
<td>DSNAMES and UNIT</td>
</tr>
<tr>
<td>GROUP</td>
<td>group level</td>
<td>GROUP_KEY, DNAME and UNIT</td>
</tr>
<tr>
<td>DBD</td>
<td>DBD level</td>
<td>GROUP_KEY, DBD_KEY, DNAME and UNIT</td>
</tr>
<tr>
<td>AREA</td>
<td>area level</td>
<td>GROUP_KEY, DBD_KEY, AREA_KEY, DNAME and UNIT</td>
</tr>
</tbody>
</table>

**Table 30  Available keywords for the LIST subcommand**

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT_AREA</td>
<td>specify area</td>
</tr>
<tr>
<td>SELECT_DBD</td>
<td>specify database</td>
</tr>
<tr>
<td>SELECT_GROUP</td>
<td>specify group</td>
</tr>
</tbody>
</table>

Sample JCL for listing selected allocation rules contained in a repository catalog at the DBD level is shown in **Figure 224**.

**Figure 224  Listing selected allocation rules**

//PFPEEXEC PGM=PFPEPR00,REGION=OM
//STEPLIB DD DSN=PFP.LOAD,DISP=SHR
// DD DSN=IMSVS.RESLIB,DISP=SHR
//PFPSYSIN DD *
PROCESS_EPR REPOSITORY_DSNAMES='PFP,PFPSEPR'
LIST DBD_ALLOCATION,SELECTGROUP=IMSA,
     SELECT_DBD=PFPDBDA
Adding allocation rules

Specify the ADD subcommand in the following format:

```
ADD _type_ALLOCATION
```

The variable _type_ refers to the level at which you want to add the allocation rule (Table 29).

Table 31 lists all keywords that can be used with the ADD subcommand to specify storage, class, statistics detail level, and retention parameters for an allocation rule.

### Table 31  Available keywords for the ADD subcommand

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>AREA_KEY</td>
<td>specify area identifier</td>
</tr>
<tr>
<td>DATACLAS</td>
<td>specify SMS data class</td>
</tr>
<tr>
<td>DBD_KEY</td>
<td>specify database identifier</td>
</tr>
<tr>
<td>DETAIL_LEVEL</td>
<td>specify level of detail (UOW or area) to be written to the repository statistics data set</td>
</tr>
<tr>
<td>DSNAME</td>
<td>specify data set name mask</td>
</tr>
<tr>
<td>EXPDT</td>
<td>specify expiration date</td>
</tr>
<tr>
<td>GROUP_KEY</td>
<td>specify group identifier</td>
</tr>
<tr>
<td>MGMTCLAS</td>
<td>specify SMS management class</td>
</tr>
<tr>
<td>RETPD</td>
<td>specify retention period</td>
</tr>
<tr>
<td>STORCLAS</td>
<td>specify SMS storage class</td>
</tr>
<tr>
<td>UNIT</td>
<td>specify physical device</td>
</tr>
<tr>
<td>VOLSER</td>
<td>specify volume serial numbers</td>
</tr>
</tbody>
</table>

Sample JCL for adding an allocation rule at the group level is shown in Figure 225. This example also shows how you can specify a retention period (90 days in this example) for statistics data sets that are added to the repository.

### Figure 225  Adding a group allocation rule for a repository

```plaintext
//PFP EXEC PGM=PFPEPR00,REGION=0M
//STEPLIB DD DSN=BMC.PFP.LOAD,DISP=SHR
// DD DSN=IMSVS.RESLIB,DISP=SHR
//PFPYSIN DD *
PROCESS_EPR REPOSITORY_DSNAME='PFP.PFPEPR'
ADD GROUP_ALLOCATION,GROUP_KEY=IMSA,
    DSNAME='PFP.&DBD.&AREA.&DATE.&TIME',
    UNIT=DISK,VOLSER=12354,RETPD=90
```
Modifying allocation rules

Specify the MODIFY subcommand in the following format:

```
MODIFY type_ALLOCATION
```

The variable `type` refers to the level at which you want to modify the allocation rule (Table 32).

Table 32 Type parameter values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Modifies</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLOBAL</td>
<td>global level allocation rule</td>
</tr>
<tr>
<td>GROUP</td>
<td>all group level allocation rules that match the criteria specified by using the SELECT_GROUP keyword</td>
</tr>
<tr>
<td>DBD</td>
<td>all DBD level allocation rules that match the criteria specified by using the SELECT_GROUP and SELECT_DBD keywords</td>
</tr>
<tr>
<td>AREA</td>
<td>all area level allocation rules that match the criteria specified by using the SELECT_GROUP, SELECT_DBD and SELECT_AREA keywords</td>
</tr>
</tbody>
</table>

`ALLOCATION` modifies all allocation rules (regardless of type) that match the selection criteria specified by using the SELECT_GROUP, SELECT_DBD and SELECT_AREA keywords.

Table 33 lists the keywords that can be used with the MODIFY subcommand to modify storage, class, statistics detail level, and retention parameters for an allocation rule.

Table 33 Available keywords for the MODIFY subcommand

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATACLAS</td>
<td>specify SMS data class</td>
</tr>
<tr>
<td>DETAIL_LEVEL</td>
<td>specify level of detail (UOW or area) to be written to the repository statistics data set</td>
</tr>
<tr>
<td>DSNAME</td>
<td>specify data set name mask</td>
</tr>
<tr>
<td>EXPDT</td>
<td>specify expiration date</td>
</tr>
<tr>
<td>MGMTCLAS</td>
<td>specify SMS management class</td>
</tr>
<tr>
<td>RETPD</td>
<td>specify retention period</td>
</tr>
<tr>
<td>SELECT_AREA</td>
<td>specify SMS storage class</td>
</tr>
<tr>
<td>SELECT_DBD</td>
<td>specify physical device</td>
</tr>
<tr>
<td>SELECT_GROUP</td>
<td>specify volume serial numbers</td>
</tr>
<tr>
<td>STORCLAS</td>
<td>specify SMS storage class</td>
</tr>
<tr>
<td>UNIT</td>
<td>specify physical storage device</td>
</tr>
<tr>
<td>VOLSER</td>
<td>specify volume serial numbers</td>
</tr>
</tbody>
</table>
Sample JCL for modifying the unit, SMS data class, and volume serial number for selected allocation rules in a repository catalog is shown in Figure 226.

**Figure 226  Modifying parameters for selected allocation rules**

```plaintext
//PFP EXEC PGM=PFPEPR00,REGION=0M
//STEP1B DD DSN=PFP.LOAD,DISP=SHR
// DD DSN=IMSVS.RESLIB,DISP=SHR
//PFPSYSIN DD *
PROCESS_EPR REPOSITORY_DSN=sname='PFP.PFPEPR'
MODIFY GROUP_ALLOCATION,SELECT_GROUP=IMSA,
SELECT_DBDB=PFPDBDA,SELECT_AREA=PFPDBDA1,
UNIT=DISK,DATACLASS=CLASS1,VOLSER=SMS001
```

**Deleting allocation rules**

Specify the DELETE subcommand in the following format:

```plaintext
DELETE type_ALLOCATION
```

The variable `type` refers to the level at which you want to delete the allocation rule (Table 32).

ALLOCATION deletes all allocation rules (regardless of type) that match the selection criteria specified using the SELECT_GROUP, SELECT_DBDB and SELECT_AREA keywords.

Table 34 lists the keywords that can be used with the DELETE subcommand to delete allocation rules and their associated parameters.

**Table 34  Available keywords for the DELETE subcommand**

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT_AREA</td>
<td>specify area</td>
</tr>
<tr>
<td>SELECT_DBDB</td>
<td>specify database</td>
</tr>
<tr>
<td>SELECT_GROUP</td>
<td>specify repository group</td>
</tr>
</tbody>
</table>

Sample JCL for deleting an allocation rule at the area level is shown in Figure 227.

**Figure 227  Deleting an area allocation rule from a repository (part 1 of 2)**

```plaintext
//PFP EXEC PGM=PFPEPR00,REGION=0M
//STEP1B DD DSN=PFP.LOAD,DISP=SHR
// DD DSN=IMSVS.RESLIB,DISP=SHR
//PFPSYSIN DD *
```
Activating repository processing

To activate repository processing when executing the analysis process, you must specify the name of the data set containing the repository catalog. The repository catalog data set name can be specified in one of the following ways:

- code the PFPEPR DD statement in the JCL
- code the REPOSITORY_DSNNAME keyword on the OPTIONS command
- include the DFSMDA member for PFPEPR in the STEPLIB DD concatenation statements

**NOTE**

If the PFPEPR DD statement is not specified in the JCL, and the REPOSITORY_DSNNAME keyword is not specified on the OPTIONS command, dynamic allocation of the repository catalog data set name will be performed by using the DFSMDA member. However, specifying the PFPEPR DD statement or the REPOSITORY_DSNNAME overrides dynamic allocation of the DFSMDA member.

Activating the repository by using JCL

An example of coding the repository catalog data set name in the JCL with the PFPEPR DD statement is shown in Figure 228.

Figure 228  Activating the repository by using JCL

```plaintext
//PFP EXEC PGM=PFPMAIN,REGION=0M
//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
//                  DD DISP=SHR,DSN=IMS.RESLIB
//PFPEPR DD DSN=BMC.PFPEPR,DISP=SHR
//areaname DD DSN=area.dataset,DISP=SHR
//PFPSYSIN DD *  
ANALYZE DBD=dbname,IAREA=areaname
```
Activating the repository by using dynamic allocation of the catalog on OPTIONS command

An example of coding the repository catalog data set name with the REPOSITORY_DSNAME keyword on the OPTIONS command is shown in Figure 229.

Figure 229  Activating the repository by using REPOSITORY_DSNAME on OPTIONS command

```plaintext
//PFP EXEC  PGM=PFPMAIN,REGION=OM
//STEPLIB DD DSN=BMC.PFP.LOAD,DISP=SHR
// areaname DD DSN=area.dataset,DISP=SHR
//PFPOPTS DD *
  OPTIONS
    REPOSITORY_DSNAME='BMC.PFPEPR'
//PFPSYSIN DD *
/*
```

Activating dynamic allocation of the repository by using DFSMDA member

To dynamically allocate the repository data set by using the dynamic allocation (DFSMDA) member for PFPEPR, use the DFSMDA member that is created during product installation. For more information, see the BMC Products for IMS Installation Guide and the Database Products for IMS Configuration Guide, or use the sample JCL that is provided in the Fast Path/EP sample library, member #PFPMDA. Then, place the dynamic allocation member in one of your STEPLIB data sets.

An example of including the DFSMDA member in the STEPLIB DD statement is shown in Figure 230.

Figure 230  Activating the repository by using DFSMDA member

```plaintext
//PFP EXEC  PGM=PFPMAIN,REGION=OM
//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD --includes DFSMDA member
// areaname DD DISP=SHR,DSN=IMS.RESLIB
//PFPSYSIN DD *
  ANALYZE DBD=dbname,IAREA=areaname
/*
```
The following items need to be considered when using the DFSMDA member to dynamically allocate the repository:

- Fast Path Analyzer/EP statistics will be added to the repository for each job that runs Fast Path Analyzer/EP. This includes jobs that are run for the reorganization, reload, and change functions.

- Additional repository maintenance should be scheduled, such as deleting entries, to manage the size of the repository catalog.

### Disabling dynamic allocation of the repository

If you need to disable (turn off) dynamic allocation of the Fast Path/EP statistics repository because of performance issues, or if statistics are not needed for specific databases, include the following DD statement in the JCL where analysis is invoked:

```plaintext
//PFPEPR DD DUMMY
```

### Specifying the repository group

Use the REPOSITORY_GROUP keyword to specify a group identifier. The group identifier is used as a part of the record key for allocation rules and statistics catalog entries contained in the repository catalog. The group identifier is used in the search for the dynamic allocation rules for creating statistics data sets, and for adding the statistics catalog entry to the repository catalog.

The REPOSITORY_GROUP keyword is intended to provide a means to distinguish between two or more database areas having identical names. For example, to distinguish between the test and production versions of a database, set `REPOSITORY_GROUP=TEST` or `REPOSITORY_GROUP=PROD`.

When an online analysis is performed, the default value for REPOSITORY_GROUP is the IMS subsystem identifier. When an offline analysis is performed, the default value for REPOSITORY_GROUP is `@@@`. 
Creating statistics data sets

When repository processing has been activated, the analysis function dynamically creates a statistics data set for each area that is analyzed. The repository catalog is searched for an allocation rule containing user-provided information for dynamically allocating a new statistics data set. An allocation rule includes fields such as the data set name mask and various allocation parameters. The product determines the DASD space required for each new statistics data set and specifies the space allocation parameters automatically.

The analysis function searches for the allocation rule to be applied in the following order:

1. It searches for an allocation rule at the area-specific level, matching the REPOSITORY_GROUP, DBD name, and area name.
2. If no matching rule is found, it searches for an allocation rule at the database-specific level, matching the REPOSITORY_GROUP and DBD name (any area name will match).
3. If no match is found, it searches for an allocation rule at the group level, matching the REPOSITORY_GROUP only (any DBD name and area name).
4. If no match is found, it obtains the global allocation rule.

Allocating the statistics data set

The name of the statistics data set is determined from the data set name mask that is defined in the selected allocation rule. If the generated data set name is NULLFILE, no statistics data set is created. If no statistics data set by the same name already exists, the data set is created and added to the operating system catalog. Other user-provided information is available in the allocation rule to assist in the allocation of new statistics data sets.

If the generated statistics data set name already exists, you can specify the action to be taken by using the REPOSITORY_OVERWRITE keyword option. If REPOSITORY_OVERWRITE=NO (the default), the original statistics data set is not overwritten and new statistics are not written for the area. Specifying REPOSITORY_OVERWRITE=YES causes the existing statistics data set to be overwritten with new data.

The sample JCL in Figure 231 shows how to set the REPOSITORY_OVERWRITE keyword through the PFPOPTS DD statement. If the statistics data set already exists for this DBD and area, the previous data is overwritten by the data from this job.
Specifying the detail level

Use the optional DETAIL_LEVEL keyword to specify the level of detail to write into the repository statistics data set. The statistics written to the repository statistics data set can provide details for each individual UOW or can be summarized into area details. The data set that is used to save summarized area details is significantly smaller than the data set that is used to save individual UOW details.

When statistics are retrieved from the repository, the level of detail (DETAIL_LEVEL = UOW or DETAIL_LEVEL = AREA) written affects the reports that can be generated. When UOW details are written, reports can be generated for a range of UOWs. When area details are written, only area reports can be generated. As a result, the STARTUOW and STOPUOW keywords cannot be used to generate reports for individual UOWs when area details are written to the repository statistics data set. For example, if you try to specify a range of UOWs when generating a report, a warning message is issued, indicating that the repository data set was written with area details, and that UOW range requests are not allowed. For more information about retrieving statistics from the repository, see “Retrieving statistics from the repository” on page 327.

The DETAIL_LEVEL keyword is an optional keyword for the ADD and MODIFY subcommands when used with allocation records. If you add an allocation record without specifying the DETAIL_LEVEL keyword, the value is set to UOW, by default. If you modify an existing allocation record without specifying the DETAIL_LEVEL keyword, there is no default value; the existing value remains the same or unchanged. For more information about using the DETAIL_LEVEL keyword, see the Fast Path/EP Series Reference Manual.
Controlling retention of repository elements

Retention of the statistics data set and retention of its associated statistics catalog entry are controlled by different methods. The statistics data set is a sequential file that will be retained according to the standards defined at your installation. Retention of the statistics catalog entries that are created by each analysis process can be controlled by managing these entries as discussed in the next section.

BMC recommends that you keep retention parameters synchronized between these two elements to prevent confusion when retrieving statistics from the repository. If the parameters for these elements are not synchronized, the following considerations apply:

- If the statistics catalog entry is present after the statistics data set has been deleted or is not available, the following items can be produced from the information that is stored in the repository statistics catalog entry:
  - Area Summary Report
  - Abbreviated version of the Free Space Analysis report
  - Information stored in the History File (For more information, see the Fast Path/EP Series Reference Manual.)

- If the statistics catalog entry has been deleted, but the statistics data set is available, the catalog entry can be recreated. For more information, see “Adding statistics catalog entries” on page 324.

Managing statistics catalog entries

The PFPEPR00 utility program offers the following subcommands that can be specified on the PROCESS_EPR command: LIST, ADD, and DELETE. When these subcommands are combined with a STATISTICS object identifier, they let you list, add, or delete statistics catalog entries from the repository catalog. Table 35 lists the statistics management and retrieval commands and subcommands that are available.

Table 35  Statistics management and retrieval commands and subcommands available with PFPEPR00 (part 1 of 2)

<table>
<thead>
<tr>
<th>Function</th>
<th>Command or subcommand</th>
<th>Available keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>selecting repository catalog for statistics processing</td>
<td>PROCESS_EPR</td>
<td>REPOSITORY_DSN AME (required)</td>
</tr>
</tbody>
</table>
Listing statistics catalog entries

The LIST subcommand is useful when you want to list the statistics catalog entries. You can also write information to a file for your own use by using the HISTORY_DDNAME keyword. For more information, see page 328.

The LIST subcommand must be specified in the following format:

```
LIST STATISTICS HISTORY_DDNAME
```

Table 36 lists all keywords that can be used with the LIST subcommand to list statistics catalog entries that are contained in the repository catalog.

Table 36 Available keywords for the LIST subcommand

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>HISTORY_DDNAME</td>
<td>Generate history file</td>
</tr>
<tr>
<td>SELECT_AREA</td>
<td>Specify area selection mask</td>
</tr>
<tr>
<td>SELECT_DATE</td>
<td>Specify date/time selection range</td>
</tr>
<tr>
<td>SELECT_DBD</td>
<td>Specify DBD selection mask</td>
</tr>
<tr>
<td>SELECT_GROUP</td>
<td>Specify group selection mask</td>
</tr>
</tbody>
</table>

The example shown in Figure 232 will list all statistics catalog entries that are contained in the repository catalog named PFP.PFPEPR.
Adding statistics catalog entries

The ADD subcommand lets you create a statistics catalog entry to register a statistics data set in a repository. This function is useful if you inadvertently ran a job with the wrong repository catalog data set name specified. Use the DSNAME keyword to specify the name of the existing statistics data set.

The ADD subcommand must be specified in the following format:

```
ADD STATISTICS
```

The example shown in Figure 233 will register the statistics data set named PFP.PFPDBDA1.STATS into the repository named PFP.PFPEPR2.

Deleting statistics catalog entries

You should coordinate the retention period for the statistics catalog entries with the options you specify in the allocation rules for the statistics data sets, such as EXPT, RETPD, GDG or SMS parameters.

The following methods are available for deleting statistics catalog entries:

- automatic deletion as part of the analysis process
- automatic deletion using the PFPEPR00 utility
- manual deletion using the Fast Path Manager/EP ISPF panels

---

**Figure 232 Sample command for writing statistics information to a data set**

```plaintext
//PFP EXEC PGM=PFPEPR00,REGION=0M
//STEPLIB DD DSN=PFP.LOAD,DISP=SHR
// DD DSN=IMSVS.RESLIB,DISP=SHR
//PFPSYSIN DD *
PROCESS_EPR REPOSITORY_DSNAME='PFP.PFPEPR'
LIST STATISTICS
```

**Figure 233 Adding a statistics catalog entry to a repository**

```plaintext
//PFP EXEC PGM=PFPEPR00,REGION=0M
//STEPLIB DD DSN=PFP.LOAD,DISP=SHR
// DD DSN=IMSVS.RESLIB,DISP=SHR
//PFPSYSIN DD *
PROCESS_EPR REPOSITORY_DSNAME='PFP.PFPEPR2'
ADD STATISTICS,DSNAME='PFP.PFPDBDA1.STATS'
```
Deleting statistics catalog entries

Chapter 11 Statistics repository facility

Automatic deletion of entries as part of the analysis process

Fast Path/EP provides the option to continually delete statistics catalog entries from the repository by specifying either or both of the following keywords on the OPTIONS command:

- REPOSITORY_RETENTION_PERIOD
- REPOSITORY_RETENTION_COUNT

The REPOSITORY_RETENTION_PERIOD keyword is processed when a new statistics catalog entry is added to the repository catalog. All previous statistics catalog entries for the same area are examined to determine their age. If the age of the entry exceeds the number of days that are specified by the REPOSITORY_RETENTION_PERIOD option, the statistics catalog entry is removed.

The REPOSITORY_RETENTION_COUNT keyword is processed when a new statistics catalog entry is added to the repository catalog. A count is taken of all statistics catalog entries for the same area. If the number of catalog entries for an area exceeds the number that is specified on the REPOSITORY_RETENTION_COUNT keyword, the oldest statistics catalog entries are removed.

NOTE
Deletion of a statistics catalog entry using the REPOSITORY_RETENTION_PERIOD keyword or the REPOSITORY_RETENTION_COUNT keyword does not cause the statistics data set to be deleted.

The sample JCL in Figure 234 shows how to set the REPOSITORY_RETENTION_PERIOD to 7 days through the PFPOPTS DD statement. This PFPOPTS DD can be included in your utility JCL for other Fast Path Offline Suite command processes, including REORGANIZE, RELOAD, and ANALYZE.

Figure 234 Specifying the REPOSITORY_RETENTION_PERIOD keyword under the PFPOPTS DD statement

```
//PFPOPTS DD *
OPTIONS
   REPOSITORY_RETENTION_PERIOD=7,
   REPOSITORY_DSNAME='PFP.PFPEPR'
/*
```

The sample JCL in Figure 235 shows how both of the repository retention limiting keywords can be specified on the same PFPOPTS DD statement. If either of the criteria is exceeded for the statistics catalog entries for a given area, (i.e., if the number of entries exceeds 5 or the entry is more than 7 days old), then the excessive record will be deleted. This PFPOPTS DD can be included in your utility JCL for other Fast Path/EP processes, including REORGANIZE, RELOAD, and ANALYZE.
Deleting statistics catalog entries

Figure 235  Specifying the REPOSITOR Y_RETENTION_PERIOD keyword and the REPOSITOR Y_RETENTION_COUNT keyword under the PFPOPTS DD statement

```
//PFPOPTS DD *
  OPTIONS
    REPOSITORY_RETENTION_PERIOD=7,
    REPOSITORY_RETENTION_COUNT=5,
    REPOSITORY_DSNAME='PFP.PFPEPR'
/*
```

**Automatic deletion of entries by using PFPEPR00**

The REPOSITORY_RETENTION_PERIOD keyword can also be specified with the DELETE subcommand to effect automatic deletion of statistics catalog entries.

**NOTE**
Deleting of the statistics catalog entry in this manner does not cause the statistics data set to be deleted.

Figure 236 shows how you can set up a job to run periodically to delete statistics catalog entries that are older than 7 days.

Figure 236  Automatic deletion of entries using PFPEPR00

```
//PFP EXEC PGM=PFPEPR00,REGION=0M
//STEPLIB DD DSN=PFP.LOAD,DISP=SHR
// DD DSN=IMSVS.RESLIB,DISP=SHR
//PFPSYSIN DD *
  PROCESS_EPR REPOSITORY_DSNAME='PFP.PFPPER'
  DELETE_STATISTICS REPOSITORY_RETENTION_PERIOD=7
```

**Manual deletion of entries**

You can use the Fast Path Manager/EP ISPF panels to manually remove statistics catalog entries from the repository catalog. You can delete the corresponding statistics data set at the same time. For more information, see “Using the Fast Path Manager/EP ISPF panels” on page 329.
Retrieving statistics from the repository

You can retrieve statistics from the repository by producing reports or by generating a history file.

Reports can be obtained by using any of the following ways:

- executing PFPEPR00 with the RETRIEVE subcommand under the PROCESS_EPR command, and execute the optional REPORT subcommand.
- executing PFPMAIN with the RETRIEVE command and the optional REPORT subcommand.
- viewing reports by using the ISPF panels that are provided by the Fast Path Manager/EP facility

Producing Reports

By specifying the RETRIEVE subcommand and the REPORT subcommand, you can retrieve and print historical reports from the repository for monitoring and analysis purposes.

Table 37 lists keywords for RETRIEVE that allow you to select the statistics catalog entries to be included in the report (or reports).

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT_GROUP</td>
<td>specify repository group</td>
</tr>
<tr>
<td>SELECT_DBD</td>
<td>specify database name (DBD)</td>
</tr>
<tr>
<td>SELECT_AREA</td>
<td>specify area</td>
</tr>
<tr>
<td>SELECT_DATE</td>
<td>specify a date and time range</td>
</tr>
<tr>
<td>SELECT_LIMIT</td>
<td>limit the selection to only the most recent entries for an area</td>
</tr>
</tbody>
</table>

Use the REPORT subcommand to control the reports to be generated. If the statistics data set was not created because the data set name was specified as NULLFILE, or if the statistics data set has been deleted or is not available, only the Area Summary Report and an abbreviated Free Space Analysis Report is produced.

The example shown in Figure 237 will generate reports from the repository named PFP.PFPEPR for the area specified by the SELECT_AREA keyword. With the exception of the Record Length Analysis Report, all standard analysis reports is generated.
Producing a history file

Fast Path Online Analyzer/EP and Fast Path Analyzer/EP provide a facility to generate a file containing historical summary records. This file can be used as input to a user-supplied reporting facility for your own internal DEDB trend analysis and planning purposes.

In addition to generating reports, you can add the HISTORY_DDNAME keyword to the RETRIEVE subcommand to specify the ddnames of DD statements specified in the JCL to which a statistical summary record is written. An example is shown in Figure 238.

You can also use the HISTORY_DDNAME keyword with the LIST STATISTICS subcommand as shown in Figure 239.
Fast Path Manager/EP ISPF panels can be used to perform the following functions:

- delete a statistics catalog entry and its corresponding statistics data set
- view online any of the reports that can be generated by using statistics stored in a selected statistics data set
- print any of the reports that can be generated by using statistics stored in a selected statistics data set

Each of these tasks requires that you access a display of the statistics catalogs entries created by previous analysis processes. You must make selections from Fast Path Manager/EP panels.
To use the Fast Path Manager/EP panels

1 From the FAST PATH MANAGER/EP main menu shown in Figure 240, select FAST PATH ANALYZER/EP.

Figure 240 FAST PATH MANAGER/EP main menu

2 From the options menu panel shown in Figure 241, select Display reports.

Figure 241 FAST PATH ANALYZER/EP options menu
The system displays the Report Options panel (Figure 242).

3 From the Report Options panel, select your sequence, print, and deletion options.

A To select the order in which you want the statistics catalog entries to be listed, select one of the Report Sequence options.

Figure 242 Report Options panel

![Report Options panel](image)

B To print selected reports, you must complete the Print Options parameters.

C Specify one of the following deletion options:

- To delete the statistics data set at the same time that you delete the statistics catalog entry, select On for the Delete Confirmation Status field.

The delete confirmation panel shown in Figure 243 is displayed for each selected statistics catalog entry. From this panel, you can delete the statistics catalog entry only or delete the statistics catalog entry and its associated statistics data set.
To delete the statistics catalog entry only, select Yes for the Delete Confirmation Status field.

4 Press Enter.

The Report List panel is displayed (Figure 244).
Figure 244  Report List panel

<table>
<thead>
<tr>
<th>Run Date/Time</th>
<th>GROUP</th>
<th>DBDname</th>
<th>AREA</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>MON 8/21/03 23.12</td>
<td>@@@@</td>
<td>DP$10</td>
<td>DP$10A0</td>
<td></td>
</tr>
<tr>
<td>THU 2/15/03 10.36</td>
<td>FPHDAMSH</td>
<td>FPHDAM01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MON 2/19/03 15.52</td>
<td>FPHDAMSH</td>
<td>FPHDAM01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TUE 2/20/03 11.44</td>
<td>FPHDAMSH</td>
<td>FPHDAM01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TUE 2/20/03 11.55</td>
<td>FPHDAMSH</td>
<td>FPHDAM01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TUE 2/20/03 11.57</td>
<td>FPHDAMSH</td>
<td>FPHDAM01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TUE 2/20/03 12.20</td>
<td>FPHDAMSH</td>
<td>FPHDAM01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TUE 2/20/03 12.26</td>
<td>FPHDAMSH</td>
<td>FPHDAM01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WED 2/21/03 21.50</td>
<td>FPHDAMSH</td>
<td>FPHDAM01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>THU 2/22/03 22.04</td>
<td>FPHDAMSH</td>
<td>FPHDAM01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MON 2/26/03 14.05</td>
<td>FPHDAMSH</td>
<td>FPHDAM01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MON 2/26/03 18.10</td>
<td>FPHDAMSH</td>
<td>FPHDAM01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MON 2/26/03 18.12</td>
<td>FPHDAMSH</td>
<td>FPHDAM01</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5  In the **Action** field, specify the action to take:

- To select the statistics catalog entries from which you want to view reports, type **S** (Display).

- To select the statistics catalog entries from which you want to print reports, type **P** (Print).

**NOTE**

When you have selected an action code, the system displays a list of reports that are available for printing or viewing.

- To delete statistics catalog entries from the repository catalog list, type **D** (Delete).

- To obtain information about the statistics data set, type **I** (Data Set).

6  Press **Enter**.
Sample repository scenarios

For scenarios that show how to use the repository commands, see Appendix C, “Sample command scenarios.”
Diagnostic procedures and tools

This appendix contains information on preparing and gathering information before contacting BMC Customer Support. This appendix discusses the following topics:

Overview ................................................................. 335
  Note software product levels ........................................ 336
  Locate generated SVC dump ........................................... 336
  Define the general problem .......................................... 338
  Provide duplication and resolution information ...................... 339
Using diagnostic commands ............................................. 341
  Using the DUMP facility ............................................... 341
Diagnosing problems with expressions ............................... 342

Overview

To assist BMC Customer Support in providing a faster response to any problem, you should have certain information available when you call. Information about software product levels and problem descriptions are required to resolve the problem.

The procedures, questions, and information requests in this appendix were prepared by BMC Customer Support to help you receive timely and quality support. Your initial effort will help resolve issues involving BMC products. You might want to copy and fill out the forms in this appendix and send them to BMC for further discussion.
Note software product levels

Gather the following basic product level information:

- FAST PATH Offline Suite product version, release, and maintenance level
- IMS version, release, and PUT level
- z/OS version

Locate generated SVC dump

Each product in the Fast Path Offline Suite will generate an SVC dump automatically when a product abend occurs. MVS will generate a notification message that the SVC dump has been created. You should locate the SVC dump that is produced by the product abend. The data set name for the SVC dump can be found in the JES messages in the JOBLOG. BMC Software might request that you supply the SVC dump with other requested documentation.

If the Fast Path product is unable to generate an SVC dump automatically following a product abend, message BMC110980E might be generated with return code 8. Table 38 lists explanations for reason codes that might be issued with the RC=8.

<table>
<thead>
<tr>
<th>Reason code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No SVC dump was requested.</td>
</tr>
<tr>
<td>1</td>
<td>An SVC dump was successfully started.</td>
</tr>
<tr>
<td>2</td>
<td>An SVC dump was suppressed because the capture phase of another SVC dump was in progress.</td>
</tr>
<tr>
<td>3</td>
<td>An SVC dump was suppressed by a request by the installation (example: DUMP=NO at IPL or CHNGDUMP SET, NODUMP).</td>
</tr>
<tr>
<td>4</td>
<td>An SVC dump was suppressed by a SLIP NODUMP command.</td>
</tr>
<tr>
<td>5</td>
<td>An SVC dump was suppressed because a SYS1.DUMP data set was not available.</td>
</tr>
<tr>
<td>6</td>
<td>An SVC dump was suppressed because an I/O error occurred during the initialization of the SYS1.DUMP data set.</td>
</tr>
<tr>
<td>8</td>
<td>An SVC dump was suppressed because an SRB could not be scheduled to activate the dump tasks in the requested address spaces.</td>
</tr>
<tr>
<td>9</td>
<td>An SVC dump was suppressed because a terminating error occurred in SVC dump before the first dump record was written.</td>
</tr>
<tr>
<td>A</td>
<td>An SVC dump was suppressed because a status stop SRB condition was detected.</td>
</tr>
<tr>
<td>B</td>
<td>An SVC dump was suppressed by Dump Analysis and Elimination (DAE).</td>
</tr>
<tr>
<td>15</td>
<td>The parameter list address is zero.</td>
</tr>
<tr>
<td>16</td>
<td>The parameter list is not a valid SVC or SNAP parameter list.</td>
</tr>
<tr>
<td>Reason code</td>
<td>Explanation</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>17</td>
<td>The caller-supplied data set is not supported.</td>
</tr>
<tr>
<td>18</td>
<td>The start address is greater than or equal to the end address in a storage list.</td>
</tr>
<tr>
<td>19</td>
<td>The caller-supplied header is longer than 100 characters.</td>
</tr>
<tr>
<td>1A</td>
<td>The caller requested a 4K buffer, but did not reserve it.</td>
</tr>
<tr>
<td>1B</td>
<td>A storage list overlaps the 4K buffer.</td>
</tr>
<tr>
<td>1C</td>
<td>The caller-supplied DCB is not valid.</td>
</tr>
<tr>
<td>1E</td>
<td>An ASID in the ASID list is syntactically not valid.</td>
</tr>
<tr>
<td>22</td>
<td>The 4K buffer was requested with an SVC dump already in progress.</td>
</tr>
<tr>
<td>25</td>
<td>A subpool ID that was not valid was specified in the subpool list.</td>
</tr>
<tr>
<td>28</td>
<td>Part of the parameter list is inaccessible.</td>
</tr>
<tr>
<td>29</td>
<td>The caller-supplied DCB is inaccessible.</td>
</tr>
<tr>
<td>2A</td>
<td>The caller-supplied storage list is inaccessible.</td>
</tr>
<tr>
<td>2B</td>
<td>The caller-supplied header data is inaccessible.</td>
</tr>
<tr>
<td>2C</td>
<td>The caller-supplied ECB is inaccessible.</td>
</tr>
<tr>
<td>2D</td>
<td>The caller’s ASID list is inaccessible.</td>
</tr>
<tr>
<td>2E</td>
<td>The caller’s SUMLIST/SUMLSTA is inaccessible.</td>
</tr>
<tr>
<td>2F</td>
<td>The caller’s SUBPLST is inaccessible.</td>
</tr>
<tr>
<td>30</td>
<td>The caller’s KEYLIST is inaccessible.</td>
</tr>
<tr>
<td>31</td>
<td>Copies of the SLIP register and PSW are inaccessible.</td>
</tr>
<tr>
<td>32</td>
<td>The caller-supplied SRB is inaccessible.</td>
</tr>
<tr>
<td>33</td>
<td>The version number in the parameter list is not valid.</td>
</tr>
<tr>
<td>34</td>
<td>The caller’s LISTD is inaccessible.</td>
</tr>
<tr>
<td>35</td>
<td>The caller’s SUMLISTL is inaccessible.</td>
</tr>
<tr>
<td>36</td>
<td>The parameter list contains conflicting parameters.</td>
</tr>
<tr>
<td>37</td>
<td>The ID is longer than 50 characters.</td>
</tr>
<tr>
<td>38</td>
<td>The ID is not addressable.</td>
</tr>
<tr>
<td>39</td>
<td>The PSWREGS area is an incorrect length.</td>
</tr>
<tr>
<td>3A</td>
<td>The PSWREGS area is not addressable.</td>
</tr>
<tr>
<td>3B</td>
<td>The symptom record is not valid.</td>
</tr>
<tr>
<td>3C</td>
<td>The symptom record is not addressable.</td>
</tr>
<tr>
<td>3D</td>
<td>The DEB for the caller-supplied DCB is inaccessible.</td>
</tr>
<tr>
<td>3E</td>
<td>SVC dump is already using the maximum amount of virtual storage (as determined by the installation, using the MAXSPACE parameter on the CHNGDUMP command) to process other dumps.</td>
</tr>
<tr>
<td>3F</td>
<td>The caller-supplied STRLIST area is inaccessible.</td>
</tr>
<tr>
<td>40</td>
<td>The caller-supplied INTOken area is inaccessible.</td>
</tr>
<tr>
<td>41</td>
<td>The caller-supplied REMOTE area is inaccessible.</td>
</tr>
<tr>
<td>42</td>
<td>The caller-supplied PROBDESC area is inaccessible.</td>
</tr>
</tbody>
</table>
Define the general problem

Table 38  Reason codes for RC=8 on message BMC110980E (part 3 of 3)

<table>
<thead>
<tr>
<th>Reason code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>43</td>
<td>The caller-supplied JOBLIST area is inaccessible.</td>
</tr>
<tr>
<td>44</td>
<td>The caller-supplied DSPLIST area is inaccessible.</td>
</tr>
<tr>
<td>45</td>
<td>The caller-supplied REMOTE area is not valid. The length of a field in the REMOTE area is specified as less than 4 bytes.</td>
</tr>
<tr>
<td>46</td>
<td>SVC dump stopped the dump because the system resources manager (SRM) detected a critical shortage of auxiliary storage.</td>
</tr>
</tbody>
</table>

Define the general problem

By answering the following questions, you will begin to refine or isolate the problem you are having:

- What product messages or IMS messages were issued before and after the problem occurred (for example, Job Log, Control Region, and Dependent Region)?

- Are any system completion codes (ABEND) issued that explain why the system abnormally terminated the product?

- Are there circumstances that caused the problem? Circumstances which might be useful information needed to diagnose the problem include
  
  — starting and stopping a message region
  — peak system stress
  — other system component failure

- Has software maintenance been applied to any software components?

- Is the problem affecting one or more of the following entities?
  
  — particular system
  — DBD
  — area
  — PSB/transaction

- How often does the problem occur? Is the problem intermittent or continuous?

- If the problem is in another vendor’s product, and Fast Path Offline Suite products might be involved, are any other software vendors such as IBM working on the problem? If so, what is the ETR number or who is the other vendor contact?
BMC Customer Support might ask you to send documentation to reproduce a problem. You might be asked to provide the following items:

- job logs and SYSOUT
- SVC dump (generated automatically by Fast Path product)
- DBDLIB and DBD source members
- ACBLIB members
- DUMP command output
- image copies of databases
- monitor reports and job history statistics (CPU, elapsed time, EXCPs)

If a problem occurs, have the symptom dump available when you call BMC Customer Support. Save the SVC dump until the problem has been resolved to your satisfaction. Customer Support will generally ask for the PSW, PSW data, and general purpose registers at time of abend. Figure 245 shows an example of a symptom dump.

**Figure 245  Sample symptom dump**

```
********************************************************************
*                                                                    *
*             BMC SOFTWARE ABEND SUMMARY                             *
*                                                                    *
* FOR ASSISTANCE, CALL BMC CUSTOMER SUPPORT                          *
*                                                                    *
* PLEASE HAVE THE FOLLOWING AVAILABLE:                               *
*                                                                    *
* 1. COMPLETE JES LOG FOR THIS JOB                                  *
* 2. PFPPRINT FOR ABENDING STEP                                    *
*                                                                    *
* SOC4 U0000  PSW 078C0000 90253EDA ILC 4 INTC 0004                *
* MOD PFSC0200 CSECT PFSC0200 EP ????????? OFFS 0706                *
* DATA 10253ED6 9680803858F0C79805EF58F0                            *
* VIRTUAL PAGE ADDRESS CAUSING EXCEPTION 00000000                   *
* ASSEMBLED ON 06/30/99 13.24                                       *
* NO FIXES APPLIED                                                  *
* REGISTERS AT ENTRY TO ABEND                                        *
* 0 102ACD68 00000000 00000004 00005FF8                             *
* 4 005F6930 102924C0 102AC380 00000000                              *
* 8 00000000 102924C0 1028F220 1020B5D0                             *
* C 902537D0 10310B78 90253AB0 10253EB0                             *
*                                                                    *
********************************************************************
```
If BMC Customer Support requests that you supply a dump, provide the appropriate accompanying documentation. Use the BMC web site to submit the dump via FTP. Contact BMC Customer Support for instructions.

**NOTE**

Abbreviated dumps (such as Abend-Aid dumps) do not provide the type of required information for problem resolution. Please ensure that a complete dump is taken.

The problem summary and documentation forms shown in **Figure 246** and **Figure 247** should be *copied*, filled out, and sent to BMC Customer Support for their understanding and resolution of your problem.

**Figure 246  Problem Summary Form (part 1 of 2)**

Fast Path Offline Suite Product

PROBLEM SUMMARY

Issue Number: ____________________  Date: __________________

Company: ____________________________________________________

Contact: ____________________________________________________

Phone:   ____________________________________________________

Email:  _____________________________________________________

----------Software Product Levels  ----------


IMS Version.Release,PUT Level:         ________.____,________

MVS Version.Release:                   ________.____,________

----------Problem Description  ----------

Product Messages:

BMC_________________________________________________________________

BMC_________________________________________________________________

BMC_________________________________________________________________

Other Messages (IMS, MVS, or JES):

_________________________________________________________________

_________________________________________________________________

System Completion Code= _____  Reason Code= ______

Circumstances that caused the problem:

_________________________________________________________________

_________________________________________________________________

______________________________________________________________

Problem Frequency:  __ Intermittent  __ Continuous

Has maintenance been applied to any software?         __  __

Describe:___________________________________

Y or N

Describe:___________________________________
Using diagnostic commands

You can use Fast Path Offline Suite commands to help with diagnostics by using the DUMP facility.

Using the DUMP facility

The DUMP command requests a dump for diagnostic information when BMC Software Customer Support instructs you to provide a dump. You will generate a dump, based on the following DUMP_TYPE keyword values:

- **ABEND** terminates the job, and generate a dump to the SYSUDUMP or SYSMDUMP data set
- **SNAP** generates a snap dump of storage areas to the data set indicated in the PFPSNAP DD
- **SVC** generates a system dump of the address space to the system dump data set

Table 39 provides information on the DUMP command set, its keywords, and parameters.
You can generate a SVC dump by command set like the following example:

```
DUMP DUMP_TYPE=SVC END
```

## Diagnosing problems with expressions

The information in this section can be used to diagnose problems with the following product functions:

- expressions that are available in the Fast Path Analyzer/EP offline data extract function

- expressions that are available in Fast Path Reorg/EP DEDB change, unload, and reload functions

The operators and operands that are used within a script or an expression are converted by the command language parser into a sequence of small operational steps. These operational steps are encoded into binary codes (called pseudo-code). This process is similar to the process that might be used by a programming language compiler to convert the source language into machine language. The generated pseudo-instructions are conceptually similar to actual machine instructions that might be executed by a CPU.

During execution of the Fast Path Offline Suite product, the pseudo-code that is generated by the parser is executed (interpreted) as required. As the pseudo-code is executed, various conditions are tested to detect problems to ensure that invalid results are not generated. When one of these tests fails, an exception report is generated, and the command function terminates. The exception report is preceded by the following message:

BMC110300C <dbdname> <areaname>: Exception code <code (name)> occurred during processing of <keyword> expression coded for the <command name> command on row <row number>, column <column number>

### Table 39  DUMP command set

<table>
<thead>
<tr>
<th>Command</th>
<th>Keyword</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUMP</td>
<td>DUMP_TYPE</td>
<td>ABEND, SNAP, SVC</td>
<td>specifies the type of dump to perform</td>
</tr>
<tr>
<td>END</td>
<td>none</td>
<td>none</td>
<td>ends command set</td>
</tr>
</tbody>
</table>

You can generate a SVC dump by command set like the following example:
The message contains the specific reason for the failure, identified by the exception code and its corresponding name. Table 40 lists the exceptions that might be produced. The most common exceptions encountered pertain to the problems with data values (Data, Overflow, Divide).

Table 40  Expression exception codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>Operation</td>
<td>invalid pseudo-instruction encountered</td>
</tr>
<tr>
<td>0005</td>
<td>Addressing</td>
<td>invalid data address encountered</td>
</tr>
<tr>
<td>0006</td>
<td>Specification</td>
<td>invalid data specification encountered</td>
</tr>
<tr>
<td>0007</td>
<td>Data</td>
<td>invalid data value encountered</td>
</tr>
<tr>
<td>0008</td>
<td>Overflow</td>
<td>value exceeds capacity of operand</td>
</tr>
<tr>
<td>0009</td>
<td>Fixed Divide</td>
<td>divide by zero (data-type F or X)</td>
</tr>
<tr>
<td>000B</td>
<td>Packed Divide</td>
<td>divide by zero (data-type P)</td>
</tr>
<tr>
<td>0010</td>
<td>Segment Trans</td>
<td>segment address not available</td>
</tr>
<tr>
<td>0012</td>
<td>Trans Specification</td>
<td>segment index invalid</td>
</tr>
<tr>
<td>0013</td>
<td>Structure Specification</td>
<td>structure index invalid</td>
</tr>
<tr>
<td>0015</td>
<td>Operand</td>
<td>run-time environment invalid</td>
</tr>
<tr>
<td>0019</td>
<td>Subscript</td>
<td>subscript out of range</td>
</tr>
<tr>
<td>0020</td>
<td>RBA Specification</td>
<td>invalid RBA value</td>
</tr>
<tr>
<td>0021</td>
<td>RBA Read</td>
<td>control interval read error</td>
</tr>
<tr>
<td>0022</td>
<td>RBA Write</td>
<td>control interval write error</td>
</tr>
<tr>
<td>0024</td>
<td>DMAC Lock</td>
<td>DMAC lock error</td>
</tr>
<tr>
<td>0028</td>
<td>Verification</td>
<td>VER function mismatch</td>
</tr>
<tr>
<td>0030</td>
<td>Stack full</td>
<td>register stack overflow</td>
</tr>
<tr>
<td>0031</td>
<td>Stack empty</td>
<td>register stack underflow</td>
</tr>
</tbody>
</table>

Message BMC110300C also identifies the failing script or expression by the keyword that is used and the position of the command containing that keyword. The remainder of the exception report provides detailed information needed for diagnosing the exact problem.

The contents of the exception report will vary, depending on the nature of the error. When possible, the product provides a specific diagnosis of the problem in a short summary report, such as the example shown in Figure 248.

Figure 248  Example exception report for common data exceptions (part 1 of 2)
Diagnosing problems with expressions

This exception report identifies the pseudo-instruction that was being processed when the error was detected. In the preceding example, an error occurred at offset 006C while executing a LOAD instruction for the data field coded as CUSTROOT.26:4P (a 4-byte packed field starting in column 26 of the root segment). Apparently, a root segment has been read that does not contain a valid packed value in this field. This is followed by a snap dump showing the contents of the segment being processed and the concatenated key of the target segment. In this example, the target segment for the expression is CURRSEGM, even though the error occurred in a field from the root segment.

For other exceptions, the Fast Path Offline Suite product produces a complete exception analysis, such as the example shown in Figure 249. In this example, a decimal divide exception has occurred during processing of the FIELDS keyword from some OUTPUT subcommand (not shown). This exception is caused when an attempt is made to divide by zero.

Figure 248  Example exception report for common data exceptions (part 2 of 2)

This exception report identifies the pseudo-instruction that was being processed when the error was detected. In the preceding example, an error occurred at offset 006C while executing a LOAD instruction for the data field coded as CUSTROOT.26:4P (a 4-byte packed field starting in column 26 of the root segment). Apparently, a root segment has been read that does not contain a valid packed value in this field. This is followed by a snap dump showing the contents of the segment being processed and the concatenated key of the target segment. In this example, the target segment for the expression is CURRSEGM, even though the error occurred in a field from the root segment.

For other exceptions, the Fast Path Offline Suite product produces a complete exception analysis, such as the example shown in Figure 249. In this example, a decimal divide exception has occurred during processing of the FIELDS keyword from some OUTPUT subcommand (not shown). This exception is caused when an attempt is made to divide by zero.

Figure 249  Example of a complete exception report (part 1 of 2)
This exception report identifies the offset of the pseudo-instruction that was being processed when the error was detected, followed by a detailed listing of all pseudo-code that was generated for the script or expression in error. While it is not necessary to understand the specific details of the pseudo-code to diagnose the problem, being able to identify the operation involved with the error might provide all required information. The example indicates that the error occurred at offset 00A2. Looking at the pseudo-code listing, you find that a DIV instruction was being processed. This is preceded by LOAD instructions for the dividend (006C), the divisor (007E), and a CONV(-1) instruction that modifies the size of the dividend in preparation for the division. The divisor is the data field coded as 18:8P (an 8-byte packed field starting in column 18 of the default segment). Apparently, in one of the input segments selected this field contains a zero value.

The exception report lists the concatenated key of the current segment (CURRSEGM), with the I/O area for that segment. The report indicates that the 8-byte field that is used as the divisor (at offset X’0011’) in this segment has a zero value.

The exception report shows the current contents of the working storage used in script or expression evaluation (if any). It also shows the current contents of the output area being generated (if any); this would not be present for a WHERE expression because it does not produce an output area.

The exception report shows the current I/O area for all other database segments used in the expression. In the example, no database segments other than the target segment are referenced.
The exception report snaps the contents of the run-time expression environment block(#EXE). The content of this storage might be useful in diagnosing certain types of exceptions but is not needed in this example.
Dynamic message modification

This appendix contains procedures on how to execute the PFPEPR00 utility program to dynamically modify eligible product messages. This appendix discusses the following topics:

Overview ................................................................. 347
  Benefits ............................................................... 348
  Customization capabilities ......................................... 348
  Activation requirement ............................................... 349
  Maintenance considerations ......................................... 349
 Message modification subcommands .................................. 349
 Dynamically modifying messages .................................... 350
  Message severity levels ............................................. 350
  Messages eligible for dynamic modification ...................... 351
  Changing severity level for a message ............................ 353
  Suppressing repetitious messages ................................. 353
 Listing message customizations ..................................... 353
 Restoring customized messages to product defaults ............... 354
 Activating message customizations .................................. 355
  Activating customizations by using JCL .......................... 355
  Activating customizations by using dynamic allocation of the catalog on OPTIONS command .............................................. 356
  Activating dynamic allocation of customizations by using DFSMDA member .................................................. 356

Overview

In addition to the analysis statistics storage and retrieval functions that are discussed in Chapter 11, “Statistics repository facility” the Fast Path Analyzer/EP statistics repository facility also lets you define and retain customizations to eligible messages that might be issued by Fast Path Offline Suite primary command processes. Eligible refers to messages that are available for customization as defined within the product. Customizations that you specify are stored in the same repository catalog that is associated with the statistics repository facility.
Benefits

You might find message customization to be useful for the following situations:

- to reduce the “nuisance factor” of a message, such as the number of times a message about segment length errors is issued for a very large area
- to reduce the severity level of a message from $E$ (error) to $W$ (warning) so that issuance of the message will not terminate processing (such as a pointer error message that would otherwise terminate an unload process)
- to increase the severity level of a condition-specific message to force an automatic snap dump to be taken in response to a detected condition within the DEDB or area

Customization capabilities

Using the PFPEPR00 utility, you can implement the following customization tasks for eligible messages that are issued by the Fast Path Offline Suite products:

- override (change) the default suffix level (informational, warning, error or critical)
- suppress issuance of the specified message by all subsequent primary command processes after reaching a specified occurrence threshold
- restore any (or all) customizations to product defaults
- list all message customizations you already specified by a previous execution of PFPEPR00 (and that are stored in the repository catalog)

Any customizations you specify for messages by using the PFPEPR00 utility will apply to all Fast Path Offline Suite primary command processes until you execute PFPEPR00 again to accomplish any of the following tasks:

- change previously specified message customizations
- add customizations for additional messages
- restore any (or all) customizations to product defaults

You can also specify message customizations on a temporary, per-job basis. For more information, see “Dynamically modifying messages” on page 55.
Activation requirement

When you have defined message customizations and have stored them in the repository catalog, you must define the repository catalog data set name in your job input to activate message customizations. For more information, see “Activating message customizations” on page 355.

Maintenance considerations

When you store message customizations in the repository and subsequently activate the repository by executing a Fast Path Offline Suite primary command set, a statistics catalog record that contains area summary statistics is written to the repository catalog. At the minimum, a statistics catalog record is written to the repository catalog for each job that activates the message customizations which are stored in the repository.

Depending on the allocation rules you have set up in your repository, these statistics catalog records and more detailed statistics data sets will accrue in the repository over a period of time when you repeatedly use the message customization feature. You might need to perform periodic maintenance to delete these statistics records.

For information about deleting the statistics catalog entries that accrue in the repository catalog when you use message customizations, see “Managing statistics catalog entries” on page 322.

Message modification subcommands

The PFPEPR00 utility program offers the following subcommands that can be specified on the PROCESS_EPR command to dynamically modify eligible product messages:

- OVERRIDE
- RESET
- LIST

The functions of available message modification commands and subcommands are listed in Table 41.
Dynamically modifying messages

The OVERRIDE subcommand is useful when you want to change the severity (suffix) level for a selected message or messages, or override default suppression levels for selected messages. Table 42 lists the functions of the keywords that can be used with the OVERRIDE subcommand.

### Table 42 Available keywords for the OVERRIDE subcommand

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MESSAGE_LEVEL</td>
<td>change default severity level for message</td>
</tr>
<tr>
<td>MESSAGE_NUMBER</td>
<td>specify product ID number of message to be customized</td>
</tr>
<tr>
<td>MESSAGE_LIMIT</td>
<td>set threshold limit for suppression of message</td>
</tr>
</tbody>
</table>

The value PFP.PFPEPR is specified on the REPOSITORY_DSNAME keyword for all sample JCL shown in this appendix. PFP.PFPEPR is the name of the default repository catalog that is created automatically during installation of a Fast Path Offline Suite product.

You must specify a separate OVERRIDE subcommand with appropriate keywords for each message that you want to customize. Multiple OVERRIDE subcommands can be specified under a single PROCESS_EPR command. In this manner, you can customize multiple messages with a single execution of the PFPEPR00 utility.

Message severity levels

Fast Path Offline Suite products issue messages with a message number followed by a letter suffix to indicate the type or severity of the message. Table 43 lists the suffix types that can be specified (changed) for a dynamically modifiable message.
Messages eligible for dynamic modification

Table 43  Fast Path Offline Suite message severity levels

<table>
<thead>
<tr>
<th>Suffix</th>
<th>Condition code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>none</td>
<td>indicates an informational message</td>
</tr>
<tr>
<td>W</td>
<td>4</td>
<td>indicates a warning message A low-severity problem exists, and processing continues.</td>
</tr>
<tr>
<td>E</td>
<td>8</td>
<td>indicates an error message A problem exists, but processing continues.</td>
</tr>
<tr>
<td>C</td>
<td>12</td>
<td>indicates a critical problem exists that must be corrected If possible, processing continues.</td>
</tr>
</tbody>
</table>

Messages eligible for dynamic modification

Table 44 lists all Fast Path Offline Suite messages which can be customized by the user. In addition to setting an issuance threshold for these messages, you can also change the default suffix (severity level) code to any of the values that are listed in Table 43.

Table 44  Fast Path Offline Suite messages available for dynamic modification  (part 1 of 2)

<table>
<thead>
<tr>
<th>Message number/ default suffix</th>
<th>Message text</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC110260E</td>
<td>Row rowNumber, column columnNumber: Randomizer module moduleName for database databaseName cannot be found in libraryName; root key retrieval will not be performed</td>
</tr>
<tr>
<td>BMC110262W</td>
<td>Index indexName: Specific Partition is not supported. PARTITION=ALL substituted</td>
</tr>
<tr>
<td>BMC111162C</td>
<td>dbdName, areaName, segmentRBA Segment length error - segment=segment length = value DMB specifies LENGTH=(max, min)</td>
</tr>
<tr>
<td>BMC111178W</td>
<td>dbdName: Change affecting all areas without IAREA=ALL</td>
</tr>
<tr>
<td>BMC111600I</td>
<td>dbdName: Count of segments selected (number) does not match count of segments written (number)</td>
</tr>
<tr>
<td>BMC111601I</td>
<td>dbdName: Count of Segments read (number) does not match count of segments written (number)</td>
</tr>
<tr>
<td>BMC111702W</td>
<td>dbdName, areaName, ddName: An error has occurred during the image copy function function; action received error code errorcode</td>
</tr>
<tr>
<td>BMC112001C</td>
<td>database,areaName: Command commandName terminated by operator</td>
</tr>
<tr>
<td>BMC115501I</td>
<td>databaseName: Segment name is being added to the output database. No occurrences of this segment will be loaded during the restructure process.</td>
</tr>
</tbody>
</table>
### Fast Path Offline Suite messages available for dynamic modification (part 2 of 2)

<table>
<thead>
<tr>
<th>Message number/ default suffix</th>
<th>Message text</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC1155031</td>
<td><code>databaseName</code>: Area name is being added to the output database. This area will be loaded during the restructure process.</td>
</tr>
<tr>
<td>BMC115540W</td>
<td><code>databaseName</code>: The DELTA IMS interface was not detected. Carefully follow the steps defined in the manual before issuing the /STA DB command.</td>
</tr>
<tr>
<td>BMC115548W</td>
<td><code>databaseName</code>: EARLY_TERMINATION was requested.</td>
</tr>
<tr>
<td>BMC121104W</td>
<td><code>ddbdName</code>, <code>areaName</code>, <code>rba</code>: Block contains an FSE at offset=offset that is not cleared to zero</td>
</tr>
<tr>
<td>BMC121110W</td>
<td><code>ddbdName</code>, <code>areaName</code>, <code>rba</code>: Block contains a SCRAP at offset=offset that is not cleared to zero</td>
</tr>
<tr>
<td>BMC121116W</td>
<td><code>ddbdName</code>, <code>areaName</code>, <code>rba</code>: Block contains an improperly formatted SCRAP at offset=offset</td>
</tr>
<tr>
<td>BMC121122W</td>
<td><code>ddbdName</code>, <code>areaName</code>, <code>rba</code>: Unexpected end of file encountered</td>
</tr>
<tr>
<td>BMC121123W</td>
<td><code>ddbdName</code>, <code>areaName</code>, <code>rba</code>: RBA mismatch - DENDRRBA=address</td>
</tr>
<tr>
<td>BMC121124W</td>
<td><code>ddbdName</code>, <code>areaName</code>, <code>rba</code>: RDF mismatch - DENDRDF=value</td>
</tr>
<tr>
<td>BMC121125W</td>
<td><code>ddbdName</code>, <code>areaName</code>, <code>rba</code>: CIDF MISMATCH - DENDCIDF=value</td>
</tr>
<tr>
<td>BMC121126W</td>
<td><code>ddbdName</code>, <code>areaName</code>, <code>rba</code>: Block flag byte invalid - DBLKFLG1=value</td>
</tr>
<tr>
<td>BMC121127W</td>
<td><code>ddbdName</code>, <code>areaName</code>, <code>rba</code>: Data block(s) exist beyond the logical end of file - DMACFBAD=block</td>
</tr>
<tr>
<td>BMC121128W</td>
<td><code>ddbdName</code>, <code>areaName</code>, <code>rba</code>: Invalid C1-type encountered in Reorg UOW.</td>
</tr>
<tr>
<td>BMC121135W</td>
<td><code>ddbdName</code>, <code>areaName</code>, <code>rba</code>: Unused bytes beyond space map are not zero</td>
</tr>
<tr>
<td>BMC121136W</td>
<td><code>ddbdName</code>, <code>areaName</code>, <code>rba</code>: Unused space map block entry at offset=offset should be zero</td>
</tr>
<tr>
<td>BMC121143W</td>
<td><code>ddbdName</code>, <code>areaName</code>, <code>rba</code>: IOVF block contains an invalid value for DBLKBTDW=value</td>
</tr>
<tr>
<td>BMC121145W</td>
<td><code>ddbdName</code>, <code>areaName</code>, <code>rba</code>: IOVF block is marked allocated but it is empty</td>
</tr>
<tr>
<td>BMC121150W</td>
<td><code>ddbdName</code>, <code>areaName</code>, <code>rba</code>: Unused SDEP block bytes beyond offset are not zero</td>
</tr>
<tr>
<td>BMC121151W</td>
<td><code>ddbdName</code>, <code>areaName</code>, <code>rba</code>: The contents of DBLKFSOF in this SDEP block are inappropriate (X'xxx')</td>
</tr>
<tr>
<td>BMC121152W</td>
<td><code>ddbdName</code>, <code>areaName</code>, <code>rba</code>: Field DBLKFSOF in this SDEP should be X'xxx': it is incorrectly set to X'xxx'</td>
</tr>
<tr>
<td>BMC121153W</td>
<td><code>ddbdName</code>, <code>areaName</code>, <code>rba</code>: Encountered an SDEP block in IMS 5.1 (or earlier) format</td>
</tr>
<tr>
<td>BMC121161W</td>
<td><code>ddbdName</code>, <code>areaName</code>, <code>rba</code>: DOVF block has the &quot;look here&quot; flag set improperly</td>
</tr>
<tr>
<td>BMC121840C</td>
<td><code>ddbdName</code>, <code>*: Bad return code code from randomizer</code></td>
</tr>
<tr>
<td>BMC121841I</td>
<td><code>ddbdName</code>, <code>*: Root key requested key randomizes to area areaName (areaNumber) that is not within the input list</code></td>
</tr>
<tr>
<td>BMC121842I</td>
<td><code>ddbdName</code>, <code>areaName</code>: Requested root key (key) not found</td>
</tr>
</tbody>
</table>
Changing severity level for a message

Figure 250 shows how the MESSAGE_NUMBER and MESSAGE_LEVEL keywords can be specified on the OVERRIDE subcommand to change the default severity level for message BMC111162 from E (the default) to W. This action will prevent the issuance of this message from terminating subsequent processes when an error is detected.

Figure 250 Changing severity level for specific message

```
//PFP EXEC PGM=PFPEPR00,REGION=0M
//STEPLIB DD DSN=PFP.LOAD,DISP=SHR
// DD DSN=IMSVS.RESLIB,DISP=SHR
//PFP SYSIN DD *
PROCESS_EPR REPOSITORY_DSNNAME='PFP.PFPEPR'
OVERRIDE MESSAGE_NUMBER=111162,
MESSAGE_LEVEL=WARNING
```

Suppressing repetitious messages

Figure 251 shows how the MESSAGE_NUMBER and MESSAGE_LIMIT keyword can be specified on the OVERRIDE subcommand to set a threshold for a selected message. For all subsequent jobs, message BMC121150W is suppressed after it is issued 15 times by the process that is defined by a primary command.

Figure 251 Specifying a suppression threshold for specific message

```
//PFP EXEC PGM=PFPEPR00,REGION=0M
//STEPLIB DD DSN=PFP.LOAD,DISP=SHR
// DD DSN=IMSVS.RESLIB,DISP=SHR
//PFP SYSIN DD *
PROCESS_EPR REPOSITORY_DSNNAME='PFP.PFPEPR'
OVERRIDE MESSAGE_NUMBER=121150,
MESSAGE_LIMIT=15
```

Listing message customizations

Figure 252 shows how the MESSAGE.Override keyword can be specified on the LIST subcommand. The PFPEPR00 utility produces a report that lists all message customizations which were stored in the repository catalog by a previous execution of the utility.
Restoring customized messages to product defaults

The MESSAGE_NUMBER keyword can be specified on the RESET subcommand to restore previous customizations to the product default for all or selected messages.

Figure 253 shows how to restore all previously specified customizations to product defaults by specifying a value of ALL on the MESSAGE_NUMBER keyword. The PFPEPR00 utility deletes all customization entries that were previously stored in the repository catalog, including severity level changes and suppression thresholds.

To restore selected customized messages to product defaults while retaining previously specified customizations for other messages, you can specify a message number on the MESSAGE_NUMBER keyword. You must specify a separate RESET subcommand and associated MESSAGE_NUMBER keyword for each message that you want to restore to product defaults.

The example shown in Figure 254 restores prior customizations to two different messages. The first RESET subcommand restores the default severity (suffix) level for the message that was originally customized by the example shown in Figure 250. The second RESET subcommand will remove the suppression threshold for the message that was originally customized by the example shown in Figure 251.

Figure 254 Restoring selected message customizations to product defaults
(part 1 of 2)
Activating message customizations

To activate message customizations when a Fast Path Offline Suite primary command is executed, you must specify the name of the repository catalog data set that contains the customizations that you specified by using the PFPEPR00 utility. The repository catalog data set name can be specified for your job input in one of the following ways:

- code the PFPEPR DD statement in the JCL
- code the REPOSITORY_DSNAME keyword on the OPTIONS command
- include the DFSMDA member for PFPEPR in the STEPLIB DD concatenation statements

**NOTE**

If the PFPEPR DD statement is not specified in the JCL, and the REPOSITORY_DSNAME keyword is not specified on the OPTIONS command, dynamic allocation of the repository catalog data set name will be performed by using the DFSMDA member. However, specifying the PFPEPR DD statement or the REPOSITORY_DSNAME overrides dynamic allocation of the DFSMDA member.

Activating customizations by using JCL

An example of coding the repository catalog data set name in the JCL with the PFPEPR DD name is shown in Figure 255.

Figure 255  Activating message customizations by using JCL

```
// DD DSN=IMSVS.RESLIB,DISP=SHR
//PFPSYSIN DD *
PROCESS_EPR REPOSITORY_DSNAME='PFP.PFPEPR'
  RESET MESSAGE_NUMBER=111162
  RESET MESSAGE_NUMBER=121150
```
Activating customizations by using dynamic allocation of the catalog on OPTIONS command

An example of coding the repository catalog data set name with the REPOSITORY_DSNAME keyword on the OPTIONS command is shown in Figure 256.

**Figure 256  Activating message customizations by using REPOSITORY_DSNAME with OPTIONS command**

```java
//PFP EXEC PGM=PFPMAIN,REGION=0M
//STEPLIB  DD DSN=BMC.PFP.LOAD,DISP=SHR
//   DD DSN=IMSVS.RESLIB,DISP=SHR
//areaname DD DSN=area.dataset,DISP=SHR
//PFPOPTS  DD *
//OPTIONS
//   REPOSITORY_DSNAME='BMC.PFPEPR'
//PFPSYSIN DD *
//   ANALYZE DBD=dbdname,IAREA=areaname
/*
```

Activating dynamic allocation of customizations by using DFSMDA member

To dynamically allocate the repository data set by using the dynamic allocation (DFSMDA) member for PFPEPR, use the DFSMDA member that is created during product installation. For more information, see the *BMC Products for IMS Installation Guide* and the *Database Products for IMS Configuration Guide*, or use the sample JCL that is provided in the Fast Path/EP sample library, member #PFPMDA. Then, place the dynamic allocation member in one of your STEPLIB data sets.

An example of including the DFSMDA member in the STEPLIB DD statement is shown in Figure 257.

**Figure 257  Activating the Repository by Using DFSMDA Member**

```java
//PFP EXEC PGM=PFPMAIN,REGION=OM
//STEPLIB  DD DISP=SHR,DSN=BMC.PFP.LOAD,DISP=SHR
//   DD DISP=SHR,DSN=IMS.RESLIB
//areaname DD DISP=SHR,DSN=area.dataset
//PFPSYSIN DD *
//   ANALYZE DBD=dbdname,IAREA=areaname
/*
```
The following items need to be considered when using the DFSMDA member to dynamically allocate the repository:

- Fast Path Analyzer/EP statistics will be added to the repository for each job that runs Fast Path Analyzer/EP. This includes jobs that are run for the reorganization, reload, and change functions.

- Additional repository maintenance should be scheduled, such as deleting entries, to manage the size of the repository catalog.
Sample command scenarios

This appendix contains sample command scenarios for the Fast Path Offline Suite product functions. This appendix includes the following topics:

How to interpret the scenarios ........................................ 360
JCL and control statement ........................................ 361
Descriptive text ......................................................... 361
Sample library JCL ...................................................... 361
Segment hierarchy for sample DEDB ................................. 362
DBD for sample DEDB .................................................. 362
Offline REORGANIZE command .................................... 363
  Simultaneously reorganize, analyze, and create image copies ........ 364
  Disable automatic analysis, place dependent segments, and stack image copies . 365
  Specify UOW selection criteria .................................... 366
  Extend IOVF .................................................................. 367
  Request repository processing, reorganize areas, and store analysis statistics in a repository ................................................. 368
Offline ANALYZE command ........................................ 369
  Analyze and detect space thresholds ............................... 370
  Analyze areas and store analysis statistics in a repository .......... 371
  Analyze areas using image copy input and create data set for area change modeling utility .................................................. 372
  Request selected reports ............................................. 373
CHANGE command ...................................................... 374
  Change area space and size parameters ........................... 376
  Change DEDB compression .......................................... 377
  Add a new segment .................................................... 379
  Increase segment lengths and initialize fields .................... 380
  Place dependent segments using LOADCTL ....................... 382
  Change area and add (build) associated indexes .................. 383
  Allocate all areas for change process using existing IDCAMS area definitions. 385
  Change area and customize space allocation across multiple volumes ...... 386
  Control SDEP allocation on a new output area ...................... 388
UNLOAD and RELOAD commands .................................... 390
  Use UNLOAD and RELOAD to increase area size and space parameters ... 392
  Use UNLOAD and RELOAD to change compression .................. 395
How to interpret the scenarios

The scenarios in this section show how to use each of the Fast Path Offline Suite primary commands. Each scenario presents JCL and an associated control statement that combines a primary command with other elements of the Fast Path Offline Suite command language to achieve a defined set of desired results. Each line of the JCL and control statement begins with a two-digit line number. Immediately following the JCL and control statement, a table provides textual descriptions of each line in the scenario. The following example shows how each scenario is presented and described.

<table>
<thead>
<tr>
<th>Scenario Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increase number of areas and initialize new fields in segments</td>
<td>397</td>
</tr>
<tr>
<td>Create a test database with a subset of data</td>
<td>400</td>
</tr>
<tr>
<td>Unload and reload a DEDB and add (build) associated indexes during reload</td>
<td>402</td>
</tr>
<tr>
<td>Unload DEDB and use FABEUR6 and FABEUR7 utilities to change segment data, then reload DEDB</td>
<td>405</td>
</tr>
<tr>
<td>Define different space allocations for reload output areas</td>
<td>407</td>
</tr>
<tr>
<td>Offline EXTRACT command</td>
<td>409</td>
</tr>
<tr>
<td>Extract and output only selected segments</td>
<td>410</td>
</tr>
<tr>
<td>Generate a subset of area data</td>
<td>411</td>
</tr>
<tr>
<td>Extract a selected segment and sort data</td>
<td>412</td>
</tr>
<tr>
<td>Modify segment layout during extract</td>
<td>413</td>
</tr>
<tr>
<td>Create a customized extract file format</td>
<td>415</td>
</tr>
<tr>
<td>INITIALIZE command</td>
<td>418</td>
</tr>
<tr>
<td>Initialize all areas for a database</td>
<td>419</td>
</tr>
<tr>
<td>Initialize and overwrite an existing output area</td>
<td>419</td>
</tr>
<tr>
<td>PROCESS_EPR command</td>
<td>421</td>
</tr>
<tr>
<td>Create and initialize a repository catalog</td>
<td>422</td>
</tr>
<tr>
<td>Add and delete allocation rules</td>
<td>423</td>
</tr>
<tr>
<td>List statistics data sets stored in repository</td>
<td>424</td>
</tr>
<tr>
<td>OPTIONS command</td>
<td>425</td>
</tr>
<tr>
<td>Request repository processing for REORGANIZE command and store analysis statistics in a repository</td>
<td>425</td>
</tr>
<tr>
<td>Request repository processing for ANALYZE command and store analysis statistics in a repository</td>
<td>427</td>
</tr>
<tr>
<td>Offline EXTEND command</td>
<td>428</td>
</tr>
<tr>
<td>Simultaneously extend IOVF, analyze, and create image copies</td>
<td>429</td>
</tr>
<tr>
<td>Extend IOVF and SDEP, disable automatic analysis, and stack image copies</td>
<td>429</td>
</tr>
<tr>
<td>Extend IOVF</td>
<td>430</td>
</tr>
</tbody>
</table>
JCL and control statement

Figure 258  JCL and control statement

```
01//PFP EXEC PGM=PFPMAIN,REGION=0M
02//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
03// DD DISP=SHR,DSN=IMSVS.RESLIB
04// DD DISP=SHR,DSN=IMSVS.DFSMDA
05//IMSACB DD DISP=SHR,DSN=IMSVS.ACBLIB
06//PFPSAMP1 DD DISP=SHR,DSN=BMCPFP.PFPSAMP.PFPSAMP1
07//PFPSAMP2 DD DISP=SHR,DSN=BMCPFP.PFPSAMP.PFPSAMP2
08//PFPSAMP3 DD DISP=SHR,DSN=BMCPFP.PFPSAMP.PFPSAMP3
09//PFPSYSIN DD *
10 REORGANIZE DBD=dbdname,POINTER_VALIDATION=FULL
11 REPORT REPORT_HEADING='ANALYSIS DURING REORG'
12 IC DSNAME='PFP.ICOPY.&DBD.&AREA(+1)',
13 UNIT=TAPE,DISP=(NEW,CATLG)
/*
```

Descriptive text

Table 45  Descriptive text for JCL

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB statements for Fast Path/EP offline execution</td>
</tr>
<tr>
<td>05</td>
<td>The IMSACB DD statement is used for database processing.</td>
</tr>
<tr>
<td>06-08</td>
<td>Define area data sets.</td>
</tr>
<tr>
<td>09</td>
<td>DD statement for Fast Path/EP control cards.</td>
</tr>
<tr>
<td>10</td>
<td>All areas of the DBD are reorganized (IAREA=ALL is default). Full (cross-reference) pointer validation is performed on all areas.</td>
</tr>
<tr>
<td>11</td>
<td>A report of the analysis is created with the specified heading.</td>
</tr>
<tr>
<td>12-13</td>
<td>An image copy is taken with a user-specified data set name. Standard storage and disposition parameters for image copy are specified.</td>
</tr>
</tbody>
</table>

Sample library JCL

The Fast Path/EP sample library contains sample JCL for each scenario that is presented in this appendix. Member $$PFPIDX contains a reference list of the scenario members.
Segment hierarchy for sample DEDB

The scenarios in this appendix are based on processes that are performed on the PFPSAMP sample DEDB, which contains three areas. Figure 259 shows a hierarchy diagram of the segments that are defined in the DBD for each area in the PFPSAMP database.

Figure 259  Segment hierarchy for sample DEDB

DBD for sample DEDB

Figure 260 provides the DBD that defines the PFPSAMP sample DEDB.

Figure 260  DBD for sample DEDB (part 1 of 2)

<table>
<thead>
<tr>
<th>DBD</th>
<th>AREA</th>
<th>SEGMENT</th>
<th>FIELD</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBD</td>
<td>NAME=PFPSAMP, ACCESS=DEDB, RMNAME=DBFHDC44</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AREA</td>
<td>DD1=PFPSAMP1.DEVICE=3380, SIZE=(4096),</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ROOT=(300,100), UOW=(30.5)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AREA</td>
<td>DD1=PFPSAMP2.DEVICE=3380, SIZE=(4096),</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ROOT=(300,100), UOW=(30.5)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AREA</td>
<td>DD1=PFPSAMP3.DEVICE=3380, SIZE=(4096),</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ROOT=(300,100), UOW=(30.5)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEGMENT</td>
<td>NAME=SEGA, BYTES=(485,18),</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PARENT=0, COMPRTN=(DPIFPRTN, DATA, INIT)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>NAME=(SEGKEY, SEQ, U), BYTES=13, START=3, TYPE=C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>NAME=FLD01, BYTES=9, START=7, TYPE=C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>NAME=FLD02, BYTES=12, START=41, TYPE=C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEGMENT</td>
<td>NAME=SDEP, BYTES=(400,25),</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>COMPRTN=(DPIFPRTN, DATA, INIT),</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PARENT=SEGA, TYPE=SEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEGMENT</td>
<td>NAME=SEGB, BYTES=(60,13),</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>COMPRTN=(DPIFPRTN, DATA, INIT),</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PARENT=SEGA, TYPE=DIR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>NAME=(SEGBK, SEQ, U), BYTES=8, START=3, TYPE=C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>NAME=FLD05, BYTES=4, START=11, TYPE=C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEGMENT</td>
<td>NAME=SEGC, BYTES=(50,13),</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PARENT=SEGB, TYPE=DIR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>NAME=(SEGCKEY, SEQ, U), BYTES=8, START=3, TYPE=C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEGMENT</td>
<td>NAME=SEGD, BYTES=(55,13),</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

362  Fast Path Offline Suite User Guide
Offline REORGANIZE command

The scenarios in this section show how to use the REORGANIZE command in conjunction with key related keywords and subcommands.

Table 46  REORGANIZE scenarios (part 1 of 2)

<table>
<thead>
<tr>
<th>Primary command/scenario task</th>
<th>Subcommand/keyword</th>
<th>Concept/process</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>REORGANIZE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simultaneously Reorganize,</td>
<td>POINTER_VALIDATION</td>
<td>■ perform one-step, concurrent maintenance and analysis tasks</td>
<td>364</td>
</tr>
<tr>
<td>Analyze, and Create Image</td>
<td>IC</td>
<td>■ create image copy of reorganized DEDB</td>
<td></td>
</tr>
<tr>
<td>Copy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REORGANIZE</td>
<td>POINTER_VALIDATION</td>
<td>■ automatic pointer validation must be turned off by user</td>
<td>365</td>
</tr>
<tr>
<td>Disable Automatic Pointer</td>
<td>LOADCTL</td>
<td>■ specify data-conditional segment placement</td>
<td></td>
</tr>
<tr>
<td>Analysis</td>
<td>SEGMENT</td>
<td>■ specify dependent segment placement based on segment count</td>
<td></td>
</tr>
<tr>
<td>Place Dependent Segments</td>
<td>LOCATION</td>
<td>■ create stacked image copies of reorganized DEDB</td>
<td></td>
</tr>
<tr>
<td>Stack Image Copies</td>
<td>LOADCTL with WHERE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>INSERT_LIMIT_COUNT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>STACK_NAME</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Simultaneously reorganize, analyze, and create image copies

Table 46  REORGANIZE scenarios (part 2 of 2)

<table>
<thead>
<tr>
<th>Primary command/scenario task</th>
<th>Subcommand/keyword</th>
<th>Concept/process</th>
<th>Page</th>
</tr>
</thead>
</table>
| REORGANIZE Specify UOW Selection Criteria | SELECT_UOW IOVF_SAVE_THRESHOLD=value IOVF_SAVE_THRESHOLD=percentage | - limit reorganization in an area to selected UOWs  
- specify different UOW selection criteria for different area  
- limit IOVF UOWs based on count or percentage | 366  
| REORGANIZE Extend IOVF | IAREA EXTEND_IOVF #UOWS IC | - limit reorg and associated tasks to one area only  
- extend IOVF during reorg  
- create output image copy of extended area | 367  
| REORGANIZE Request Repository Processing Reorganize Areas Store Analysis Statistics in Repository | OPTIONS command with following keywords: REPOSITORY_DSNAME REPOSITORY_GROUP REPOSITORY_RETENTION_COUNT REPOSITORY_OVERWRITE | - activate repository processing by using OPTIONS command  
- specify name of repository and repository group  
- request overwriting of existing statistics data set  
- specify maximum number of repository statistics catalog entries that will remain recorded within the repository catalog  
- store analysis statistics gathered during reorganization process in repository | 368  

Simultaneously reorganize, analyze, and create image copies

Figure 261  JCL to simultaneously reorganize, analyze, and create image copies

```
01   //PFP EXEC PGM=PFPMAIN,REGION=OM
02   //STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
03   // DD DISP=SHR,DSN=IMSVS.RESLIB
04   // DD DISP=SHR,DSN=IMSVS.DFSMDA
05   // IMSACB DD DISP=SHR,DSN=IMSVS.ACBLIB
06   //PFPSAMP1 DD DISP=OLD,DSN=BMCPFPP.FFPSAMP.PFPSAMP1
07   //PFPSAMP2 DD DISP=OLD,DSN=BMCPFPP.FFPSAMP.PFPSAMP2
08   //PFPSAMP3 DD DISP=OLD,DSN=BMCPFPP.FFPSAMP.PFPSAMP3
09   //PFPSYSIN DD *
10  REORGANIZE DBD=PFPSAMP.POINTER_VALIDATION=FULL
11   IC DSNAMES='PFP.ICOPY.&DBD.&AREA(+1)',
12   UNIT=TAPE,DISP=(NEW,CATLG)
/*
```
Table 47  Descriptive text for JCL to simultaneously reorganize, analyze, and create image copy

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB DD statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05</td>
<td>IMSACB DD statement used for database processing.</td>
</tr>
<tr>
<td>06-08</td>
<td>Area data set names.</td>
</tr>
<tr>
<td>09</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>10</td>
<td>A complete analysis (full pointer validation) of all areas is performed on the DEDB.</td>
</tr>
<tr>
<td>11-12</td>
<td>An output image copy is created in the same job step as the REORGANIZE. The DBD and area name are substituted in the image copy data set name to create a unique name for each area.</td>
</tr>
</tbody>
</table>

Disable automatic analysis, place dependent segments, and stack image copies

Figure 262  JCL to disable automatic analysis, place dependent segments, and stack image copies

```plaintext
01//PFP EXEC PGM=PFPMAIN,REGION=0M
02//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
03//          DD DISP=SHR,DSN=IMSVS.RESLIB
04//          DD DISP=SHR,DSN=IMSVS.DFSMDA
05//          DD DISP=SHR,DSN=BMC.DPK.LOAD
06//$$DPICDS DD DISP=SHR,DSN=PFP.DPK.DPICDS
07//$$DPITBL DD DISP=SHR,DSN=PFP.DPK.DPITBL
08//IMSACB DD DISP=SHR,DSN=IMSVS.ACBLIB
09//PFPSAMP1 DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP1
10//PFPSAMP2 DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP2
11//PFPSAMP3 DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP3
12//PFPSYSIN DD *
13 REORGANIZE DBD=PFPSAMP,AREA=ALL,
14    POINTER_VALIDATION=NONE,
15    SELECT_UOW=ALL
16 LOADCTL SEGMENT=(SEGB,DEPENDENTS),
17    LOCATION=IOVF,WHERE=(FLD05 EQ 'INAC')
18 LOADCTL SEGMENT=SEGF,LOCATION=IOVF,
19    INSERT_LIMIT_COUNT=1
20 LOADCTL SEGMENT=SEGG,LOCATION=DOVF
21 IC STACK_NAME=NAME1,
22    UNIT=TAPE,DISP=(NEW,CATLG),
23    DSNAMES='PFP.ICOPY.&DBD.&AREA(+1)' */
```
Specify UOW selection criteria

In the JCL shown in Figure 263, only two of the areas in the PFPSAMP database are reorganized.

Figure 263 JCL to specify UOW selection criteria (part 1 of 2)

```
01 //PFP EXEC PGM=PFPMAIN,REGION=0M
02 //STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
03 // DD DISP=SHR,DSN=IMSVS.RESLIB
04 // DD DISP=SHR,DSN=IMSVS.DFSMDA
05 //IMSCAB DD DISP=SHR,DSN=IMSVS.ACLIB
06 //PFPSAMP1 DD DISP=OLD,DSN=BMCFPP.PFPSAMP.PFPSAMP1
07 //PFPSAMP2 DD DISP=OLD,DSN=BMCFPP.PFPSAMP.PFPSAMP2
08 //PFPSYIN DD *
09 REORGANIZE DBD=PFPSAMP, IAREA=PFPSAMP1,10
10 SELECT_UOW=IOVF, IOVF_SAVE_THRESHOLD=
11 REORGANIZE DBD=PFPSAMP, IAREA=PFPSAMP2,
```
Before executing this control statement, you must ensure that adequate space is present on the primary volume of the target area data set to accommodate the request for additional UOWs. If the necessary space is not present, allocate volumes to the area data set by using IDCAMS ALTER ADDVOLUME.

**NOTE**

Be sure to revise the DBD to increase IOVF so that subsequent offline processes (such as CHANGE or UNLOAD/RELOAD) do not use the old ACB and decrease the amount of IOVF space.

---

**Extend IOVF**
Request repository processing, reorganize areas, and store analysis statistics in a repository

Table 50 Descriptive text for JCL to extend IOVF

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB DD statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05</td>
<td>IMSACB DD statement used for database processing.</td>
</tr>
<tr>
<td>06</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>07</td>
<td>Area PFPSAMP1 is identified on the REORGANIZE statement, and is dynamically allocated from its dynamic allocation member in the IMSVS.DFSMDA data set in the STEPLIB concatenation.</td>
</tr>
<tr>
<td>08</td>
<td>If you have a license for the BMC Software Fast Path Analyzer/EP product, quick (checksum) pointer validation is performed on area PFPSAMP1 by default. This pointer validation is performed by default, even if you do not specify it explicitly.</td>
</tr>
<tr>
<td>09</td>
<td>Adding 50 UOWs to IOVF increases space for IOVF. Increasing space in this manner allows you to perform a DBD change of the UOW value at a later time.</td>
</tr>
<tr>
<td>10-11</td>
<td>During the area extend, an output image copy is taken simultaneously using standard storage and disposition parameters. The DBD and area name are substituted in the image copy data set name to create a unique name for each area.</td>
</tr>
</tbody>
</table>

Request repository processing, reorganize areas, and store analysis statistics in a repository

Figure 264 JCL to extend IOVF (part 2 of 2)

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>PFPMAIN EXEC PGM=PFPMAIN,REGION=0M</td>
</tr>
<tr>
<td>02</td>
<td>STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD</td>
</tr>
<tr>
<td>03</td>
<td>DD DISP=SHR,DSN=IMSVS.RESLIB</td>
</tr>
<tr>
<td>04</td>
<td>DD DISP=SHR,DSN=IMSVS.DFSMDA</td>
</tr>
<tr>
<td>05</td>
<td>IMSACB DD DISP=SHR,DSN=ACBLIB.OLD</td>
</tr>
<tr>
<td>06</td>
<td>PFPSAMP1 DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP1</td>
</tr>
<tr>
<td>07</td>
<td>PFPSAMP2 DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP2</td>
</tr>
<tr>
<td>08</td>
<td>PFPSAMP3 DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP3</td>
</tr>
<tr>
<td>09</td>
<td>PFPOPTS DD *</td>
</tr>
<tr>
<td>10</td>
<td>OPTIONS</td>
</tr>
<tr>
<td>11</td>
<td>REPOSITORY_DSNAMES='BMC.PFP.PFPEPR',</td>
</tr>
<tr>
<td>12</td>
<td>REPOSITORY_GROUP='TEST',</td>
</tr>
</tbody>
</table>

Fast Path Offline Suite User Guide
Offline ANALYZE command

The scenarios in this section show how to use the offline ANALYZE command in conjunction with key related keywords and subcommands.

Figure 265  JCL to request repository processing, reorganize areas, and store analysis statistics in a repository (part 2 of 2)

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05</td>
<td>IMSACB DD statement used for database processing.</td>
</tr>
<tr>
<td>06-08</td>
<td>Specifies the data base areas to be reorganized and analyzed.</td>
</tr>
<tr>
<td>09</td>
<td>PFPOPTS DD for Fast Path/EP OPTIONS statement (line 10).</td>
</tr>
<tr>
<td>10</td>
<td>OPTIONS command is required to specify subsequent repository keywords.</td>
</tr>
<tr>
<td>11</td>
<td>Defines the repository to be used.</td>
</tr>
<tr>
<td>12</td>
<td>Identifies the group from which the area’s allocation rules should be derived.</td>
</tr>
<tr>
<td>13</td>
<td>Specifies that up to five statistics data sets for these data base areas will be stored in the repository catalog at any one time.</td>
</tr>
<tr>
<td>14</td>
<td>Indicates that if the statistics data set name defined for these areas exists, it will be overwritten with the new statistics data set created by this job.</td>
</tr>
<tr>
<td>16</td>
<td>All areas of the database are reorganized. By default, quick (checksum) pointer validation is performed on all areas in the DEDB. This pointer validation creates the statistics for database PFPSAMP to be stored in the repository that is identified in line 11.</td>
</tr>
</tbody>
</table>
Table 52  ANALYZE scenarios

<table>
<thead>
<tr>
<th>Primary command/scenario task</th>
<th>Subcommand/keyword</th>
<th>Concept/process</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANALYZE</td>
<td>POINTERS_VALIDATION</td>
<td>■ request full pointer validation</td>
<td>370</td>
</tr>
<tr>
<td>Analyze and Detect Space</td>
<td>THRESHOLD</td>
<td>■ detect various freespace/overflow thresholds</td>
<td></td>
</tr>
<tr>
<td>Thresholds</td>
<td>DOVF_FREESPACE_PERCENT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IOVF_FREESPACE_PERCENT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RAA_FREESPACE_PERCENT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RAP_OVERFLOW_PERCENT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANALYZE</td>
<td>PFPEPR DD statement</td>
<td>■ specify repository name using PFPEPR DD statement</td>
<td>371</td>
</tr>
<tr>
<td>Analyze Areas</td>
<td>OPTIONS command</td>
<td>■ activate repository processing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>REPOSITORY_OVERWRITE</td>
<td>■ request overwriting of existing statistics data set</td>
<td></td>
</tr>
<tr>
<td>STORE Analysis Statistics in</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Repository</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANALYZE</td>
<td>INPUT DSN_MASK</td>
<td>■ request full pointer validation</td>
<td>372</td>
</tr>
<tr>
<td>Analyze Areas Using Image</td>
<td>MODEL_DDNAME</td>
<td>■ analyzer input can come from an image copy</td>
<td></td>
</tr>
<tr>
<td>Copy Input</td>
<td></td>
<td>■ dynamic allocation of image copy data sets</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ create output data set for Area Change Modeling Utility</td>
<td></td>
</tr>
<tr>
<td>ANALYZE</td>
<td>REPORT_DEFAULT</td>
<td>■ disable default generation of all reports</td>
<td>373</td>
</tr>
<tr>
<td>Request Selected Reports</td>
<td>FREESPACE_ANALYSIS</td>
<td>■ request generation of selected reports only (override default)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IOVF_SPACE_ANALYSIS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 266  JCL to analyze and detect space thresholds (part 1 of 2)

```
01//PFP EXEC PGM=PFPMAIN,REGION=OM
02//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
03//     DD DISP=SHR,DSN=IMSVS.RESLIB
04//     DD DISP=SHR,DSN=IMSVS.DFSMDA
05//IMSAO B DD DISP=SHR,DSN=IMSVS.ACBLIB
06//FPFSAMP1 DD DISP=SHR,DSN=BMCPFP.FPFSAMP.FPFSAMP1
07//FPFSAMP2 DD DISP=SHR,DSN=BMCPFP.FPFSAMP.FPFSAMP2
08//FPFSAMP3 DD DISP=SHR,DSN=BMCPFP.FPFSAMP.FPFSAMP3
09//FPFSYSIN DD *
10 ANALYZE DBD=PFPSAMP,POINTER_VALIDATION=FULL
11 THRESHOLD
12 DOVF_FREESPACE_PERCENT=10,
13 IOVF_FREESPACE_PERCENT=15.
```
Analyze areas and store analysis statistics in a repository

Statistics from the analysis process are stored in the repository. The PFPEPR DD statement indicates the repository catalog to be used. This job also overwrites any previously existing statistics data set by the same name. An allocation rule has been previously set up in the repository to create statistics data sets using the data set name mask ‘PFP.&IMSID.&DBD.&AREA’. For more information, see Chapter 11, “Statistics repository facility.”

Figure 267  JCL to analyze areas and store analysis statistics in a repository (part 1 of 2)

```plaintext
01//PFPP EXEC PGM=PFPPMAIN,REGION=0M
02//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
03// DD DISP=SHR,DSN=IMSVS.RESLIB
04// DD DISP=SHR,DSN=IMSVS.DFSMDA
05//IMSACB  DD DISP=SHR,DSN=IMSVS.ACBLIB
06//PFPSAMP1  DD DISP=SHR,DSN=BMCPFPP.PFPSAMP.PFPSAMP1
07//PFPSAMP2  DD DISP=SHR,DSN=BMCPFPP.PFPSAMP.PFPSAMP2
08//PFPSAMP3  DD DISP=SHR,DSN=BMCPFPP.PFPSAMP.PFPSAMP3
```
Analyze areas using image copy input and create data set for area change modeling utility

Figure 267  JCL to analyze areas and store analysis statistics in a repository (part 2 of 2)

Table 54  Descriptive text for JCL to analyze areas and store analysis statistics in a repository

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB DD statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05</td>
<td>IMSACB DD statement used for database processing.</td>
</tr>
<tr>
<td>06-08</td>
<td>Area data set names.</td>
</tr>
<tr>
<td>09</td>
<td>PFPEPR DD statement specifies the name of the repository to be used.</td>
</tr>
<tr>
<td>10-12</td>
<td>PFPOPTS DD is required for the subsequent OPTIONS command, which is used to define default parameters for the repository. In this case, it defines a temporary override for the repository (REPOSITORY_OVERWRITE=YES).</td>
</tr>
<tr>
<td>12</td>
<td>This job overwrites any previously existing statistics data set which has the same name as that generated by the data set name mask that is specified for the DSNAME allocation rule in the repository catalog.</td>
</tr>
<tr>
<td>13</td>
<td>End of OPTIONS control card input.</td>
</tr>
<tr>
<td>14</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>15</td>
<td>Analysis is performed for all areas. Quick (checksum) pointer validation is performed on all areas in the DEDB by default.</td>
</tr>
</tbody>
</table>

Analyze areas using image copy input and create data set for area change modeling utility

Figure 268  JCL to analyze areas using image copy input and create data set for area change modeling utility (part 1 of 2)
Figure 268  JCL to analyze areas using image copy input and create data set for area change modeling utility (part 2 of 2)

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB DD statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05</td>
<td>IMSACB DD statement used for database processing.</td>
</tr>
<tr>
<td>06-07</td>
<td>The output for the Fast Path Analyzer/EP Area Change Modeling Utility is written to the DD name MODELER.</td>
</tr>
<tr>
<td>08</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>09</td>
<td>Analysis is performed on all areas.</td>
</tr>
<tr>
<td>10</td>
<td>Full (cross-reference) pointer validation is performed on all areas in the DEDB. Specifying POINTER_VALIDATION=FULL is required for the Area Change Modeling Utility.</td>
</tr>
<tr>
<td>11</td>
<td>Specifies dynamic allocation of the input image copy data sets.</td>
</tr>
<tr>
<td>12</td>
<td>Specifies the DD name in the JCL for an output data set to receive data for the Area Change Modeling Utility.</td>
</tr>
</tbody>
</table>

Table 55  Descriptive text for JCL to analyze areas using image copy input and create data set for area change modeling utility

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB DD statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05</td>
<td>IMSACB DD statement used for database processing.</td>
</tr>
<tr>
<td>06-07</td>
<td>The output for the Fast Path Analyzer/EP Area Change Modeling Utility is written to the DD name MODELER.</td>
</tr>
<tr>
<td>08</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>09</td>
<td>Analysis is performed on all areas.</td>
</tr>
<tr>
<td>10</td>
<td>Full (cross-reference) pointer validation is performed on all areas in the DEDB. Specifying POINTER_VALIDATION=FULL is required for the Area Change Modeling Utility.</td>
</tr>
<tr>
<td>11</td>
<td>Specifies dynamic allocation of the input image copy data sets.</td>
</tr>
<tr>
<td>12</td>
<td>Specifies the DD name in the JCL for an output data set to receive data for the Area Change Modeling Utility.</td>
</tr>
</tbody>
</table>

Request selected reports

Figure 269  JCL to request selected reports

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>PFP EXEC PGM=PFPMAIN,REGION=OM</td>
</tr>
<tr>
<td>02</td>
<td>STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD</td>
</tr>
<tr>
<td>03</td>
<td>DD DISP=SHR,DSN=IMSVS.RESSLIB</td>
</tr>
<tr>
<td>04</td>
<td>DD DISP=SHR,DSN=IMSVS.DFSMDA</td>
</tr>
<tr>
<td>05</td>
<td>IMSACB DD DISP=SHR,DSN=IMSVS.ACBLIB</td>
</tr>
<tr>
<td>06</td>
<td>PFPSAMP1 DD DISP=SHR,DSN=BMCPFP.PFPSAMP.PFPSAMP1</td>
</tr>
<tr>
<td>07</td>
<td>PFPSYSIN DD</td>
</tr>
<tr>
<td>08</td>
<td>ANALYZE DBD=PFPSAMP,AREA=PFPSAMP1</td>
</tr>
<tr>
<td>09</td>
<td>REPORT REPORT_DEFAULT=NO,</td>
</tr>
<tr>
<td>10</td>
<td>FREESPACE_ANALYSIS=YES,</td>
</tr>
<tr>
<td>11</td>
<td>IOVF_SPACE_ANALYSIS=YES</td>
</tr>
</tbody>
</table>
|         | /*

Appendix C  Sample command scenarios
### CHANGE command

The scenarios in this section show how to use the CHANGE command in conjunction with key related keywords and subcommands.

#### Table 56  Descriptive text for JCL to request selected reports

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB DD statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05</td>
<td>IMSACB DD statement used for database processing.</td>
</tr>
<tr>
<td>06</td>
<td>Area data set name.</td>
</tr>
<tr>
<td>07</td>
<td>PFFSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>08</td>
<td>Only the PFPsamp1 area is analyzed. Quick (checksum) pointer validation is performed on area PFPsamp1 by default.</td>
</tr>
<tr>
<td>09</td>
<td>Sets report generation default to NO (no reports are generated unless specified by name).</td>
</tr>
<tr>
<td>10</td>
<td>The Free Space Analysis Report is generated.</td>
</tr>
<tr>
<td>11</td>
<td>The IOVF Space Analysis Report is generated.</td>
</tr>
</tbody>
</table>

#### Table 57  CHANGE scenarios (part 1 of 2)

<table>
<thead>
<tr>
<th>Primary command/scenario task</th>
<th>Subcommand/keyword</th>
<th>Concept/process</th>
<th>Page</th>
</tr>
</thead>
</table>
| CHANGE                        | IAREA / OAREA I with STACK_NAME AREA statements from DBD | ■ change the size of UOW, RAA, and CI for areas based on changed AREA statements in DBD  
■ create stacked image copies of reorganized DEDB | 376  |
| CHANGE                        | IAREA / OAREA EXPAND SEGM statements from DBD  | ■ apply added/changed compression routines for selected segments based on changed SEGM statements in DBD 
■ expand compressed data using newly defined routines | 377  |
| CHANGE                        | IAREA / OAREA SDEP_PROCESS POINTER_VALIDATION IC | ■ add segment to all areas based on new SEGM statement in DBD 
■ specify processing for defined SDEPs  
■ create output image copy of extended area | 379  |
<table>
<thead>
<tr>
<th>Primary command/scenario task</th>
<th>Subcommand/keyword</th>
<th>Concept/process</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHANGE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Increase Segment Lengths</td>
<td>IAREA / OAREA EXPAND INPUT_DSN_MASK OUTPUT_DSN_MASK OUTPUT SEGMENT FIELDS</td>
<td>increase segment length for specified segments</td>
<td>380</td>
</tr>
<tr>
<td>Initialize Fields</td>
<td>LOADCTL SEGMENT LOCATION LOADCTL with WHERE</td>
<td>disable automatic pointer validation</td>
<td>382</td>
</tr>
<tr>
<td>Place Dependent Segments</td>
<td>multiple IX with INDEX</td>
<td>rebuild all associated indexes during CHANGE</td>
<td>383</td>
</tr>
<tr>
<td>using LOADCTL</td>
<td>Allocate All Areas for Change Process using Existing IDCAMS Definitions</td>
<td>allocate output area using existing IDCAMS definitions</td>
<td>385</td>
</tr>
<tr>
<td>Change an Area</td>
<td>Allocate Area and Customize Space Allocation Across Multiple Volumes</td>
<td>divide RAA across multiple volumes</td>
<td>386</td>
</tr>
<tr>
<td>Add (Build) Associated Indexes</td>
<td>Control SDEP Allocation on a New Change Output Area</td>
<td>divide SDEP across multiple volumes</td>
<td>388</td>
</tr>
</tbody>
</table>
Change area space and size parameters

The UOW, ROOT and CI size values are increased for selected areas. Changes are made in the DBD to increase the CI, UOW and ROOT values for area PFPSAMP1, and to increase the CI size for area PFPSAMP2. Area PFPSAMP3 is included because the DBD specifies a generalized randomizer, and roots will randomize due to the changes made.

**NOTE**

The new ACB with the DBD changes must be gened before executing the CHANGE control statement. Also, the output area data sets which reflect the DBD changes must be defined before executing the CHANGE control statement.

Figure 270  JCL to change area space and size parameters

```
01//PFP EXEC PGM=PFPMAIN,REGION=0M
02//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
03// DD DISP=SHR,DSN=IMSVS.RESLIB
04// DD DISP=SHR,DSN=IMSVS.DFSMDA
05//OLDACB DD DISP=SHR,DSN=IMSVS.ACBLIB.OLD
06//IMSACB DD DISP=SHR,DSN=IMSVS.ACBLIB.NEW
07//IAREA001 DD DISP=SHR,DSN=BMCPFP.PFPSAMP.PFPSAMP1
08//IAREA002 DD DISP=SHR,DSN=BMCPFP.PFPSAMP.PFPSAMP2
09//IAREA003 DD DISP=SHR,DSN=BMCPFP.PFPSAMP.PFPSAMP3
10//PFPSAMP1 DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP1.NEW
11//PFPSAMP2 DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP2.NEW
12//PFPSAMP3 DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP2.NEW
13//PFPSYSIN DD *
14 CHANGE DBD=PFPSAMP,
15  POINTER_VALIDATION=QUICK,
16  SDEP_PROCESS=LOGICAL,
17  IAREA=
18  (PFPSAMP1,
19    PFPSAMP2,
20     PFPSAMP3),
21 OAREA=
22  (PFPSAMP1,
23    PFPSAMP2,
24    PFPSAMP3)
25 IC DSNAME='PFP.ICOPY.&DBD.&AREA(+1)',
26  STACK_NAME=STACK1,UNIT=TAPE,DISP=(NEW,CATLG)
/*
```
The compression routine is changed for three segments based on changes made in the DBD.

**NOTE**

The new ACB with the DBD changes must be genned before executing the CHANGE control statement. Also, the output area data sets which reflect the DBD changes must be defined before executing the CHANGE control statement.
Change DEDB compression

**Figure 271 JCL to change DEDB compression**

```plaintext
01//PFP EXEC PGM=PFPMAIN,REGION=0M
02//STEPLIB DD DSN=BMC.PFP.LOAD,DISP=SHR
03// DD DSN=IMSVS.DFSMDA,DISP=SHR
04// DD DSN=IMSVS.RESLIB,DISP=SHR <-- DFSCMPXO
05// DD DSN=BMC.DPK.LOAD,DISP=SHR <-- DPIFPRTN
06//$$DPICDS DD DISP=SHR,DSN=PFP.DPK.DPICDS
07//$$DPITBL DD DISP=SHR,DSN=PFP.DPK.DPITBL
08//OLDACB DD DSN=IMSVS.ACBLIB.OLD,DISP=SHR
09//IMSA CB DD DSN=IMSVS.ACBLIB.NEW,DISP=SHR
10//IAREA001 DD DSN=PFP.ICOPY.PFPSAMP.PFPSAMP1(0),DISP=SHR
11//IAREA002 DD DSN=PFP.ICOPY.PFPSAMP.PFPSAMP2(0),DISP=SHR
12//IAREA003 DD DSN=PFP.ICOPY.PFPSAMP.PFPSAMP2(0),DISP=SHR
13//PFPSAMP1 DD DSN=BMCPF P.PFPSAMP.PFPSAMP1.NEW,DISP=OLD
14//PFPSAMP2 DD DSN=BMCPF P.PFPSAMP.PFPSAMP2.NEW,DISP=OLD
15//PFPSAMP3 DD DSN=BMCPF P.PFPSAMP.PFPSAMP3.NEW,DISP=OLD
16//PFPSYSIN DD *
17 CHANGE DBD=PFPSAMP,
18 SDEP_PROCESS=LOGICAL,
19 IAREA=ALL,
20 OAREA=ALL,
21 EXPAND=YES
*/
```

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-4</td>
<td>EXEC and STEPLIB statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>04</td>
<td>Compression routines specified in IMSACB DD are used on output areas.</td>
</tr>
<tr>
<td>05</td>
<td>Compression routines specified in OLDACB DD are used on input areas.</td>
</tr>
<tr>
<td>06-07</td>
<td>DD statements that define data sets for the BMC Software DATA PACKER®/IMS product. These data sets are required because the segment data must be expanded in order to apply the new compression routines.</td>
</tr>
<tr>
<td>08</td>
<td>The DBD from the OLDACB DD statement is used for input areas.</td>
</tr>
<tr>
<td>09</td>
<td>The DBD from the IMSACB DD statement is used for output areas.</td>
</tr>
<tr>
<td>10-12</td>
<td>Inputs are image copy data sets.</td>
</tr>
<tr>
<td>13-15</td>
<td>Output area data sets</td>
</tr>
<tr>
<td>16</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>17</td>
<td>Defines DEDB name for CHANGE input/output.</td>
</tr>
<tr>
<td>18</td>
<td>Because the DEDB contains SDEPs, SDEP processing must be specified.</td>
</tr>
<tr>
<td>19</td>
<td>IAREA specifies the input areas for the IAREAxxx DD statements.</td>
</tr>
<tr>
<td>20</td>
<td>OAREA specifies the area names for the PFPSAMPx DD statements.</td>
</tr>
</tbody>
</table>
Add a new segment

A new segment (SEGH) has been added to the DBD at the end of the hierarchical structure. The CHANGE command must be executed to establish the prefix pointer for the added segment.

**NOTE**

The new ACB with the DBD changes must be genned before executing the CHANGE control statement. Also, the output area data sets which reflect the DBD changes must be defined before executing the CHANGE control statement.

### Table 59 Descriptive text for JCL to change DEDB compression (part 2 of 2)

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>19 and 21</td>
<td>All compressed segments in the input are expanded with the compression routines (DPIFPRTN) specified in OLDACB data set. The three changed segments are:</td>
</tr>
<tr>
<td></td>
<td>SEGM NAME=SEGB,BYTES=(100,20),PARENT=SEGA,COMPRTN=(DPIFPRTN,DATA,INIT)</td>
</tr>
<tr>
<td></td>
<td>SEGM NAME=SEGC,BYTES=(100,20),PARENT=SEGB,COMPRTN=(DPIFPRTN,DATA,INIT)</td>
</tr>
<tr>
<td></td>
<td>SEGM NAME=SEGD,BYTES=(100,20),PARENT=SEGB,COMPRTN=(DPIFPRTN,DATA,INIT)</td>
</tr>
<tr>
<td>20 and 21</td>
<td>Output areas are compressed using changed/added compression routines specified in the IMSACB data set:</td>
</tr>
<tr>
<td></td>
<td>For SEGB, the compression exit name in the DBD is changed from DPIFPRTN (BMC Software DATA PACKER®/IMS product) to DFSCMPX0 (IBM compression utility).</td>
</tr>
<tr>
<td></td>
<td>For SEGC, the DFSCMPX0 compression exit name is added.</td>
</tr>
<tr>
<td></td>
<td>For SEGD, compression is removed. The DFSCMPX0 compression exit name has been deleted from the SEGM statement for SEGD.</td>
</tr>
<tr>
<td></td>
<td>SEGM NAME=SEGB,BYTES=(100,20),PARENT=SEGA,COMPRTN=(DFSCMPX0,DATA,INIT)</td>
</tr>
<tr>
<td></td>
<td>SEGM NAME=SEGC,BYTES=(100,20),PARENT=SEGB,COMPRTN=(DFSCMPX0,DATA,INIT)</td>
</tr>
<tr>
<td></td>
<td>SEGM NAME=SEGD,BYTES=(100,20),PARENT=SEGB</td>
</tr>
</tbody>
</table>

### Figure 272 JCL to add a new segment (part 1 of 2)

```
01//PFPP EXEC   PGM=PFPPMAIN,REGION=OM
02//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
03// DD DISP=SHR,DSN=IMSVS.RESLIB
04// DD DISP=SHR,DSN=IMSVS.DFSMDA
05//OLDACB DD DSN=IMSVS.ACBLIB.OLD,DISP=SHR
06//IMSACB DD DSN=IMSVS.ACBLIB.NEW,DISP=SHR
07//IAREA001 DD DSN=BMCRIPT.PFPsamp.PFPsamp1,DISP=SHR
08//IAREA002 DD DSN=BMCRIPT.PFPsamp.PFPsamp2,DISP=SHR
09//IAREA003 DD DSN=BMCRIPT.PFPsamp.PFPsamp3,DISP=SHR
10//PFPsamp1 DD DISP=OLD,DSN=BMCRIPT.PFPsamp.PFPsamp1.NEW
11//PFPsamp2 DD DISP=OLD,DSN=BMCRIPT.PFPsamp.PFPsamp2.NEW
12//PFPsamp3 DD DISP=OLD,DSN=BMCRIPT.PFPsamp.PFPsamp3.NEW
13//PFPPSYSIN DD *
```
Increase segment lengths and initialize fields

**Figure 272  JCL to add a new segment (part 2 of 2)**

```plaintext
14  CHANGE DBD=PFPSAMP,
15    IAREA=ALL,
16    OAREA=ALL,
17    SDEP_PROCESS=LOGICAL,
18    POINTER_VALIDATION=FULL
19    IC DSNAME='PFP.ICOPY.&DBD.&AREA(+1)',
20    UNIT=TAPE,DISP=(NEW,CATLG)
/*
```

**Table 60  Descriptive text for JCL to add a new segment**

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-4</td>
<td>EXEC and STEPLIB statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05</td>
<td>The DBD from the OLDACB DD statement is used for input areas.</td>
</tr>
<tr>
<td>06</td>
<td>The DBD from the IMSACB DD statement is used for output areas, which contains the newly defined segment: Segment NAME=SEGH,BYTES=(100,20),PARENT=SEGA</td>
</tr>
<tr>
<td>10,11,12</td>
<td>OAREA specifies the output area names for the PFPSAMPx DD statements.</td>
</tr>
<tr>
<td>13</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>14</td>
<td>Defines DEDB name for CHANGE input/output.</td>
</tr>
<tr>
<td>15</td>
<td>IAREA specifies the input areas for the IAREAxxx DD statements.</td>
</tr>
<tr>
<td>16</td>
<td>OAREA specifies the output area names for the PFPSAMPx DD statements.</td>
</tr>
<tr>
<td>17</td>
<td>Because the DEDB contains SDEPs, SDEP processing must be specified.</td>
</tr>
<tr>
<td>18</td>
<td>Full pointer validation is performed simultaneously while taking the image copy.</td>
</tr>
<tr>
<td>19-20</td>
<td>Image copies are taken of the changed database.</td>
</tr>
</tbody>
</table>

**Increase segment lengths and initialize fields**

**Figure 273  JCL to increase segment lengths and initialize fields (part 1 of 2)**

```plaintext
01//PFP EXEC  PGM=PFPMAIN,REGION=0M
02//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
03//          DD DISP=SHR,DSN=IMSVS.RESLIB
04//          DD DISP=SHR,DSN=IMSVS.DFSMDA
05//          DD DISP=SHR,DSN=BMC.DPK.LOAD
06//$$DPICDS DD DISP=SHR,DSN=PFPP.DPK.DPICDS
07//$$DPIBBL DD DISP=SHR,DSN=PFPP.DPK.DPIBBL
08//OLDACB    DD DSN=IMSVS.ACBLIB.OLD,DISP=SHR
09//IMSACB    DD DSN=IMSVS.ACBLIB.NEW,DISP=SHR
10//PFPSYSIN  DD *
11  CHANGE DBD=PFPSAMP,
12    EXPAND=YES,
13    SDEP_PROCESS=LOGICAL.
```
Increase segment lengths and initialize fields

Table 61  Descriptive text for JCL to increase segment lengths and initialize fields

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05-07</td>
<td>DD statements that define data sets for the BMC DATA PACKER/IMS product. These data sets are required because data is being added in the compressed portion of the segments.</td>
</tr>
<tr>
<td>08</td>
<td>The DBD from the OLDACB DD statement is used for input areas.</td>
</tr>
<tr>
<td>09</td>
<td>The DBD from the IMSACB DD statement is used for the output areas.</td>
</tr>
<tr>
<td>10</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>11</td>
<td>Defines DEDB name for CHANGE input/output. The CHANGE statement processes all input areas (IAREA=ALL is default). If you have a license for the BMC Software Fast Path Analyzer/EP product, quick (checksum) pointer validation is performed on all areas by default.</td>
</tr>
<tr>
<td>12</td>
<td>EXPAND=YES keyword is required because SEGD is a compressed segment and must be expanded before adding the new field.</td>
</tr>
<tr>
<td>13</td>
<td>Because the DEDB contains SDEPs, SDEP processing must be specified.</td>
</tr>
<tr>
<td>14</td>
<td>INPUT_DSN_MASK keyword defines the data set mask to be used to dynamically allocate input data sets.</td>
</tr>
<tr>
<td>15</td>
<td>OUTPUT_DSN_MASK keyword defines the data set mask to be used to dynamically allocate output area data sets.</td>
</tr>
<tr>
<td>16 and 18</td>
<td>OUTPUT subcommands identify the modified segments. All other database segments are written to the output areas, but are not changed.</td>
</tr>
</tbody>
</table>
| 16-17    | For SEGC, a new 6-byte character field is inserted after the key. This new field is initialized to zeroes, and the previous data is shifted to a position after the new field. The values that are specified on the FIELDS keyword copies bytes starting in column 3 from the input, 6 bytes of zeroes, followed by the input data starting at column 11, to the end of the input data. Max length is increased by 6 bytes:  
  ``
  SEGM  NAME=SEGC,BYTES=(56,13)
  ``` |
| 18-19    | For SEGD, a 5-byte packed decimal field is added at the end of the segment and initialized to zeroes. Values that are specified on the FIELDS keyword copy the full input segment (starting at column 3, to the end of the data), followed by the 5-byte initialized field. Max length is increased by 5 bytes:  
  ```
  SEGM  NAME=SEGD,BYTES=(60,13)
  ``` |
Place dependent segments using LOADCTL

To maintain segment placement that is executed by this scenario, you must continue to use the same LOADCTL subcommands for any subsequent reorganization, change, or reload processing.

**NOTE**

The new ACB with the DBD changes must be generated before executing the CHANGE control statement. Also, the output area data sets which reflect the DBD changes must be defined before executing the CHANGE control statement.

---

**Figure 274  JCL to place dependent segments using LOADCTL**

```plaintext
01//PFP EXEC   PGM=PFPMAIN,REGION=OM
02//STEPLIB    DD DISP=SHR,DSN=BMC.PFP.LOAD
03//          DD DISP=SHR,DSN=IMSVS.RESLIB
04//          DD DISP=SHR,DSN=IMSVS.DFSMDA
05//          DD DISP=SHR,DSN=BMC.DPK.LOAD
06//$$DPICDS   DD DISP=SHR,DSN=PFP.DPK.DPICDS
07//$$DPITBL   DD DISP=SHR,DSN=PFP.DPK.DPITBL
08//OLDACB     DD DSN=IMSVS.ACBLIB,DISP=SHR
09//IMSACB     DD DSN=IMSVS.ACBLIB,DISP=SHR
10//IAREA001   DD DSN=BMCPFP.PFPSAMP.PFPSAMP1,DISP=SHR
11//IAREA002   DD DSN=BMCPFP.PFPSAMP.PFPSAMP2,DISP=SHR
12//IAREA003   DD DSN=BMCPFP.PFPSAMP.PFPSAMP3,DISP=SHR
13//PFPSAMP1   DD DSN=BMCPFP.PFPSAMP.PFPSAMP1.NEW,DISP=OLD
14//PFPSAMP2   DD DSN=BMCPFP.PFPSAMP.PFPSAMP2.NEW,DISP=OLD
15//PFPSAMP3   DD DSN=BMCPFP.PFPSAMP.PFPSAMP3.NEW,DISP=OLD
16//PFPSYSIN   DD *
17 CHANGE DBD=PFPSAMP,
18  SDEP_PROCESS=LOGICAL,
19  POINTER_VALIDATION=NONE
20 LOADCTL SEGMENT=(SEGB,DEPENDENTS),
21  LOCATION=IOVF,WHERE=(FLD05 EQ 'INAC')
22 LOADCTL SEGMENT=SEGF,LOCATION=IOVF,
23  INSERT_LIMIT_COUNT=1
24 LOADCTL SEGMENT=SEGG,LOCATION=DOVF
/*
```

**Table 62  Descriptive text for JCL to place dependent segments using LOADCTL (part 1 of 2)**

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05-07</td>
<td>DD statements that define data sets for the BMC Software DATA PACKER®/IMS product. These data sets are required because SEGB is a compressed segment and must be expanded internally to read FLD05.</td>
</tr>
<tr>
<td>21</td>
<td>DD statements that define data sets for the BMC Software DATA PACKER®/IMS product. These data sets are required because SEGB is a compressed segment and must be expanded internally to read FLD05.</td>
</tr>
</tbody>
</table>
The functionality in this scenario requires that you have a license for the BMC Software Fast Path Indexer/EP product. The structure of the PFPSAMP database is being changed by the addition of a new area, and two indexes are being added. A separate IX subcommand is specified for each index to be built.

NOTE
The new ACB with the DBD changes must be genned before executing the CHANGE control statement. Also, the output area data sets and index data sets which reflect the DBD changes must be defined before executing the CHANGE control statement.

Figure 275   JCL to change area and add (build) associated indexes (part 1 of 2)

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>PFP EXEC PGM=PFPMAIN, REGION=0M</td>
</tr>
<tr>
<td>02</td>
<td>STEPLIB DD DISP=SHR, DSN=BMC.PFP.LOAD</td>
</tr>
<tr>
<td>03</td>
<td>STEPLIB DD DISP=SHR, DSN=IMSVS.RESLIB</td>
</tr>
<tr>
<td>04</td>
<td>STEPLIB DD DISP=SHR, DSN=IMSVS.DFSMDA</td>
</tr>
<tr>
<td>05</td>
<td>DD DISP=SHR, DSN=BMC.DPK.LOAD</td>
</tr>
<tr>
<td>06</td>
<td>DPICDS DD DISP=SHR, DSN=PFP.DPK.DPICDS</td>
</tr>
<tr>
<td>07</td>
<td>DPITBL DD DISP=SHR, DSN=PFP.DPK.DPITBL</td>
</tr>
<tr>
<td>08</td>
<td>OLDACB DD DSN=IMSVS.ACBLIB, DISP=SHR</td>
</tr>
<tr>
<td>09</td>
<td>IMSACB DD DSN=IMSVS.ACBLIB.NEW, DISP=SHR</td>
</tr>
<tr>
<td>10</td>
<td>OLDLIB DD DSN=IMSVS.PFXLIB, DISP=SHR</td>
</tr>
</tbody>
</table>
Table 63  Descriptive text for JCL to change area and add (build) associated indexes (part 1 of 2)

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05-07</td>
<td>DD statements that define data sets for the BMC Software DATA PACKER®/IMS product. These data sets are required because the index PFPINDEX2 is built using the SRCH field FLDO1, which is part of a compressed segment. The compressed segment must be expanded internally before building the index.</td>
</tr>
<tr>
<td>08</td>
<td>The DBD from the OLDACB DD statement is used for input areas.</td>
</tr>
<tr>
<td>09, 15-18</td>
<td>The DBD from the IMSACB DD statement is used for output areas. An area is added (PFPSAMP4), which increases the total number of areas to 4. Two indexes are added to the SEGA statement, as indicated by the PFXCHILD and PFXXDFLD statements. The following changes are made in the DBD: AREA DD1=PFPSAMP4,DEVICE=3380,CISIZE=(4096),... SEGm NAME=SEGA PFXCHILD NAME=(XSEG1,PFPINDEX1),... PFXXDFLD NAME=XFLD,SRCH=SEGAKEY,... PFXCHILD NAME=(XSEG2,PFPINDEX2),... PFXXDFLD NAME=XFLD,SRCH=FLDO1,SUBSEQ=SEGAKEY,...</td>
</tr>
<tr>
<td>10</td>
<td>OLDLIB DD refers to the old index registration library.</td>
</tr>
<tr>
<td>11</td>
<td>PFXLIB DD refers to the new index registration library.</td>
</tr>
<tr>
<td>12-14</td>
<td>All input area data sets.</td>
</tr>
<tr>
<td>15-18</td>
<td>All output area data sets.</td>
</tr>
<tr>
<td>19, 24</td>
<td>Specifies output index allocation and build for PFPINDEX1.</td>
</tr>
<tr>
<td>20, 25</td>
<td>Specifies output index allocation and build for PFPINDEX2</td>
</tr>
<tr>
<td>21</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
</tbody>
</table>
Allocate all areas for change process using existing IDCAMS area definitions

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>Defines DEDB name for CHANGE input/output. The CHANGE statement processes all input areas (IAREA=ALL is default) and output areas (OAREA=ALL is default), including the new area PFPSAMP4. If you have a license for the BMC Software Fast Path Analyzer/EP product, quick (checksum) pointer validation is performed on all areas by default.</td>
</tr>
<tr>
<td>23</td>
<td>Because the DEDB contains SDEPs, SDEP processing must be specified.</td>
</tr>
<tr>
<td>24</td>
<td>Build index database PFPIPDX1.</td>
</tr>
<tr>
<td>25</td>
<td>Build index database PFPIPDX2. The SRCH field FLD01 is part of the compressed data for segment SEGA. Segment SEGA is expanded internally to ensure that the index PFPIPDX2 is built with expanded data.</td>
</tr>
</tbody>
</table>

NOTE
This technique can also be used on a RELOAD or INITIALIZE command.

Figure 276 JCL to allocate all areas for change process using existing IDCAMS area definitions

```plaintext
01//PFP EXEC PGM=PFPMAIN,REGION=0M
02//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
03// DD DISP=SHR,DSN=IMSVS.RESLIB
04// DD DISP=SHR,DSN=IMSVS.DFSMDA
05//OLDACB DD DISP=SHR,DSN=IMSVS.ACBLIB.OLD
06//IMSACB DD DISP=SHR,DSN=IMSVS.ACBLIB
07//IAREA001 DD DISP=SHR,DSN=BMCPFP.PFPSAMP.PFPSAMP1.OLD
08//IAREA002 DD DISP=SHR,DSN=BMCPFP.PFPSAMP.PFPSAMP2.OLD
09//IAREA003 DD DISP=SHR,DSN=BMCPFP.PFPSAMP.PFPSAMP3.OLD
10//PFPSYSIN DD *
11 CHANGE DBD=PFPSAMP,OAREA=ALL
12 SDEP_PROCESS=LOGICAL
13 ALLOCATE OAREA=ALL,
14 ACTUATE=(IDCAMS,DSN='IMSVS.IDCSRC(&AREA)')
/*
Change area and customize space allocation across multiple volumes

Because no extent size parameter is specified for RAA or IOVF in this scenario, extent size is determined automatically. Because the SPACE keyword is not specified, IDCAMS determines the extent size in tracks or cylinders to accommodate RAA and IOVF.

**NOTE**

This technique can also be used on a RELOAD or INITIALIZE command.

---

### Table 64  Descriptive text for JCL to allocate all areas for change process using existing IDCAMS area definitions

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB DD statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05</td>
<td>The DBD from the OLDACB DD statement is used for input areas.</td>
</tr>
<tr>
<td>06</td>
<td>The DBD from the IMSACB DD statement is used for output areas.</td>
</tr>
<tr>
<td>07-09</td>
<td>Input area data sets.</td>
</tr>
<tr>
<td>10</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>11</td>
<td>Defines DEDB name for CHANGE input/output. CHANGE statement processes all input areas (IAREA=ALL is default) and output areas (OAREA=ALL is specified). The output areas are dynamically allocated. If you have a license for the BMC Software Fast Path Analyzer/EP product, quick (checksum) pointer validation is performed on all areas by default.</td>
</tr>
<tr>
<td>12</td>
<td>Because the DEDB contains SDEPs, SDEP processing must be specified.</td>
</tr>
<tr>
<td>13</td>
<td>All output areas of PFPSAMP are allocated.</td>
</tr>
<tr>
<td>14</td>
<td>Specifies a data set name mask with the IDCAMS parameter on the ACTUATE keyword. All output areas are allocated from an existing IDCAMS data set definition. The &amp;AREA substitution variable is used to resolve the PDS member that specifies the VSAM cluster definition.</td>
</tr>
</tbody>
</table>

---

### Figure 277  JCL to change area and customize space allocation across multiple volumes (part 1 of 2)

```
01//PFP EXEC PGM=PFPMAIN,REGION=0M  
02//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD, 
03//     DD DISP=SHR,DSN=IMSVS.RESLIB  
04//     DD DISP=SHR,DSN=IMSVS.DFSMDA  
05//OLDACB DD DSN=IMSVS.ACBLIB.OLD,DISP=SHR  
06//IMSACB DD DSN=SHR,DSN=IMSVS.ACBLIB  
07//IAREA001 DD DISP=SHR,DSN=PFPSAMP.ICOPY.PFPSAMP.PFPSAMP1(0)  
08//PFPSYSIN DD *  
09 CHANGE DBD=PFPSAMP,IAREA=PFPSAMP1,OAREA=PFPSAMP1  
10 SDEP_PROCESS=LOGICAL
```
Change area and customize space allocation across multiple volumes

Figure 277  JCL to change area and customize space allocation across multiple volumes (part 2 of 2)

Table 65  Descriptive text for JCL to change area and customize space allocation across multiple volumes

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>EXEC and STEPLIB DD statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05</td>
<td>The DBD from the OLDACB DD statement is used for input areas.</td>
</tr>
<tr>
<td>06</td>
<td>The DBD from the IMSACB DD statement is used for output areas.</td>
</tr>
<tr>
<td>07</td>
<td>Input image copy data set</td>
</tr>
<tr>
<td>08</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>09</td>
<td>Defines DEDB name for CHANGE input/output. Area PFPSAMP1 is changed. The input area is identified on the IAREA001 DD statement, and is an image copy. The output area is dynamically allocated. If you have a license for the BMC Software Fast Path Analyzer/EP product, quick (checksum) pointer validation is performed on all areas by default.</td>
</tr>
<tr>
<td>10</td>
<td>Because the DEDB contains SDEPs, SDEP processing must be specified.</td>
</tr>
<tr>
<td>11</td>
<td>Output area PFPSAMP1 is allocated.</td>
</tr>
<tr>
<td>12</td>
<td>ACTUATE keyword parameters rename the current VSAM cluster according to the data set name mask.</td>
</tr>
<tr>
<td>13</td>
<td>If an error occurs, the EROPT=ABORT parameter terminates ALLOCATE subcommand processing</td>
</tr>
<tr>
<td>14</td>
<td>The VOLCNT parameter is included on the CONFIGURE_RAA keyword to specify that the root addressable portion be spanned evenly over two volumes.</td>
</tr>
<tr>
<td>15</td>
<td>The VOLCNT parameter is included on the CONFIGURE_IOVF keyword to specify that the independent overflow portion of the output area be wholly contained on one volume.</td>
</tr>
<tr>
<td>16</td>
<td>The VOLCNT parameter is included on the CONFIGURE_SDEP keyword to specify that the SDEP portion of the output area be wholly contained on one volume.</td>
</tr>
<tr>
<td>17</td>
<td>The VOLSER keyword identifies the candidate volumes that are used for the VSAM cluster definition. An asterisk (*) indicates that specific VOLSERs are obtained from SMS.</td>
</tr>
<tr>
<td>18</td>
<td>The STORCLAS keyword specifies the SMS storage class to be assigned to the VSAM cluster.</td>
</tr>
</tbody>
</table>

Figure 278 shows how the RAA portion will be divided evenly across two volumes, based on the value that is specified for the VOLCNT parameter on the CONFIGURE_RAA keyword. IOVF will be contained on one volume as specified by the VOLCNT parameter value on the CONFIGURE_IOVF keyword.
Control SDEP allocation on a new output area

Because the CONFIGURE_RAA and CONFIGURE IOVF keywords are not specified in this scenario, the control statement automatically calculates the combined size of the RAA and IOVF portions of the output area and allocates them to one volume.

Figure 279  JCL to control SDEP allocation on a new output area

```
01//PFP EXEC PGM=PFPMAIN,REGION=OM
02//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD.
03// DD DISP=SHR,DSN=IMSVS.RESLIB
04// DD DISP=SHR,DSN=IMSVS.DFSMDA
05//OLDACB DD DSN=IMSVS.ACBLIB.OLD,DISP=SHR
06//IMSACB DD DISP=SHR,DSN=IMSVS.ACBLIB
07//IAREA002 DD DISP=SHR,DSN=BMCPFP.PFPSAMP.PFPSAMP2.OLD
08//PFPSYSIN DD *
09 CHANGE DBD=PFPSAMP,IAREA=PFPSAMP2,OAREA=PFPSAMP2
10 SDEP_PROCESS=LOGICAL
11 ALLOCATE OAREA=PFPSAMP2,
12 CONFIGURE_SDEP=(VOLCNT=2,500),
13 SPACE=CYL,
14 VOLSER=(sysda,sysda,sysda)
/*
```
### Table 66  Descriptive text for JCL to control SDEP allocation on a new output area

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>EXEC and STEPLIB DD statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05</td>
<td>The DBD from the OLDACB DD statement is used for input areas.</td>
</tr>
<tr>
<td>06</td>
<td>The DBD from the IMSACB DD statement is used for output areas.</td>
</tr>
<tr>
<td>07</td>
<td>Input area data set.</td>
</tr>
<tr>
<td>08</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>09</td>
<td>Defines DEDB name for CHANGE input/output. Area PFPSAMP2 is changed. If you have a license for the BMC Software Fast Path Analyzer/EP product, quick (checksum) pointer validation is performed on all areas by default.</td>
</tr>
<tr>
<td>10</td>
<td>Because the DEDB contains SDEPs, SDEP processing must be specified.</td>
</tr>
<tr>
<td>11</td>
<td>Output area PFPSAMP2 is allocated.</td>
</tr>
<tr>
<td>12</td>
<td>The VOLCNT parameter is included on the CONFIGURE_SDEP keyword to specify that the sequential dependent portion of the output area be spanned over two volumes. The second parameter on the CONFIGURE_SDEP keyword specifies that the size of each extent be 500 cylinders.</td>
</tr>
<tr>
<td>13</td>
<td>The SPACE keyword specifies cylinders as the allocation unit.</td>
</tr>
<tr>
<td>14</td>
<td>The VOLSER keyword identifies three volumes (one volume to contain RAA and IOVF, and two volumes to contain SDEP as requested by the CONFIGURE_SDEP keyword).</td>
</tr>
</tbody>
</table>

Figure 280 shows how the VOLCNT parameters that are specified on the CONFIGURE_SDEP keyword will spread the SDEP portion of the area across two volumes, each with 500 cylinders of space. The entire area is spread across three volumes as requested by the VOLSER keyword.
The scenarios in this section show how to use the UNLOAD and RELOAD commands in conjunction with key related keywords and subcommands. Most of these scenarios show both UNLOAD control card input and RELOAD control card input in order to demonstrate the respective JCL requirements that are necessary to successfully execute these processes.
Table 67  UNLOAD and RELOAD scenarios (part 1 of 2)

<table>
<thead>
<tr>
<th>Primary command/scenario task</th>
<th>Subcommand/keyword</th>
<th>Concept/process</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNLOAD/RELOAD Increase Area Space/Size Parameters</td>
<td>IAREA / OAREA AREA DD statements SDEP_PROCESS (on UNLOAD) SORT (on RELOAD) IC (on RELOAD)</td>
<td>■ change the size of UOW, RAA, and CI for areas ■ specify processing for defined SDEPs ■ request image copy to be taken on reload</td>
<td>392</td>
</tr>
<tr>
<td>UNLOAD/RELOAD Change DEDB Compression</td>
<td>IAREA / OAREA EXPAND (on UNLOAD) COMPRESS (on RELOAD) IC (on RELOAD) SEGMENT statements from DBD</td>
<td>■ apply added/changed compression routines for selected segments based on changed SEGMENT statements in DBD ■ expand compressed data using newly defined routines ■ request image copy to be taken on reload</td>
<td>395</td>
</tr>
<tr>
<td>UNLOAD/RELOAD Increase Number of Areas Initialize New Fields in Segments</td>
<td>IAREA / OAREA INPUT_DSN_MASK OUTPUT_DSN_MASK OUTPUT SEGMENT FIELDS POINTER_VALIDATION EXPAND (on RELOAD) SDEP_PROCESS IC</td>
<td>■ request image copy to be taken on reload ■ specify allocation parameters for image copy ■ increase segment length for specified segments ■ expand compressed segments ■ request NO pointer validation on reload ■ dynamically allocate input and output data sets ■ specify processing for defined SDEPs</td>
<td>397</td>
</tr>
<tr>
<td>UNLOAD/RELOAD Create a Test Database with a Subset of Data</td>
<td>INPUT_DSN_MASK INCLUDE SEGMENT SAMPLE_LIMIT SAMPLE_INTERVAL OFILECTL SDEP_PROCESS ERROR_THRESHOLD</td>
<td>■ dynamic allocation of input areas ■ include sampling of root segments of interval and limit criteria ■ specify processing for defined SDEPs ■ define and dynamically allocate output files</td>
<td>400</td>
</tr>
<tr>
<td>UNLOAD/RELOAD Unload and Reload a DEDB Add (Build) Associated Indexes During Reload</td>
<td>INPUT_DSN_MASK SDEP_PROCESS OUTPUT_DSN_MASK OFILECTL IX with INDEX IC</td>
<td>■ dynamic allocation of input and output areas ■ specify processing for defined SDEPs ■ define output unload file ■ add (build) multiple associated indexes during reload process</td>
<td>402</td>
</tr>
</tbody>
</table>
Use UNLOAD and RELOAD to increase area size and space parameters

The UOW, ROOT and CI size values are increased for selected areas. Changes are made in the DBD to increase the CI, UOW and ROOT values for area PFPSAMP1, and to increase the CI size for area PFPSAMP2.

**NOTE**

The new ACB with the DBD changes must be genned before executing the UNLOAD control statement.

---

**Table 67  UNLOAD and RELOAD scenarios (part 2 of 2)**

<table>
<thead>
<tr>
<th>Primary command/scenario task</th>
<th>Subcommand/keyword</th>
<th>Concept/process</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNLOAD/RELOAD</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| Unload DEDB, Use FABEUR6 and FABEUR7 Utilities to Change Segment Data, then Reload DEDB | SDEP_PROCESS OFILECTL with OAREA IC with STACK_NAME | ▪ specify processing for defined SDEPs  
▪ write all unloaded areas to one output file  
▪ Use FABEUR6/7 utilities to make changes to segment data  
▪ create stacked image copy for each reloaded area | 405 |
| RELOAD                        |                   |                 |      |
| Define Different Space Allocations for Reload Output Areas | ALLOCATE with OAREA ACTUATE SPACE / VOLSER | ▪ define allocation for multiple output areas  
▪ define different space allocation for each output area | 407 |

---

**Use UNLOAD and RELOAD to increase area size and space parameters**

**Figure 281  JCL to use UNLOAD and RELOAD to increase area size and space parameters (part 1 of 2)**

```
01//PFPUNLD EXEC PGM=PFPMAIN,REGION=0M
02//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
03// DD DISP=SHR,DSN=IMSVS.RESLIB
04// DD DISP=SHR,DSN=IMSVS.DFSMDA
05//IMSACB DD DISP=SHR,DSN=IMSVS.ACBLIB
06//NEWACB DD DISP=SHR,DSN=IMSVS.ACBLIB.NEW
07//PFPSAMP1 DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP1
08//PFPSAMP2 DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP2
09//PFPSAMP3 DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP3
10//OAREA001 DD DSN=PFP.PFPSAMP1.UNLOAD,
11// DISP=(,CATLG,DELETE),UNIT=DASD,SPACE=(CYL,(65,5),RLSE)
12//OAREA002 DD DSN=PFP.PFPSAMP2.UNLOAD,
13// DISP=(,CATLG,DELETE),UNIT=DASD,SPACE=(CYL,(65,5),RLSE)
14//OAREA003 DD DSN=PFP.PFPSAMP3.UNLOAD,
15// DISP=(,CATLG,DELETE),UNIT=DASD,SPACE=(CYL,(65,5),RLSE)
```
Use UNLOAD and RELOAD to increase area size and space parameters

Table 68 Descriptive text for JCL to use UNLOAD and RELOAD to increase area size and space parameters (part 1 of 2)

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05-07</td>
<td>The DBD from the IMSACB DD statement is used to define the input areas.</td>
</tr>
<tr>
<td>06-15</td>
<td>The DBD from the NEWACB DD statement is used to define the output unload files.</td>
</tr>
<tr>
<td>07-09</td>
<td>Define input areas to be unloaded.</td>
</tr>
<tr>
<td>10-15</td>
<td>The unloaded data is written to the specified file.</td>
</tr>
<tr>
<td>16</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
</tbody>
</table>
### Table 68  Descriptive text for JCL to use UNLOAD and RELOAD to increase area size and space parameters (part 2 of 2)

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>Defines DEDB name for UNLOAD processing. Because the DEDB contains SDEPs, SDEP processing must be specified.</td>
</tr>
<tr>
<td>18</td>
<td>IAREA specifies the input areas for the PFPSAMPx DD statements.</td>
</tr>
<tr>
<td>19</td>
<td>Area PFPSAMP1 is input using the old DBD size parameters: AREA DD1=PFPSAMP1, SIZE=4096, UOW=(30,5), ROOT=(300,100)</td>
</tr>
<tr>
<td>20</td>
<td>Area PFPSAMP2 is input using the old DBD size parameters: AREA DD1=PFPSAMP2, SIZE=4096</td>
</tr>
<tr>
<td>21</td>
<td>Area PFPSAMP3 is input. No changes are made to area PFPSAMP3. Because the DEDB uses a generalized randomizer, however, all areas must be processed.</td>
</tr>
<tr>
<td>22</td>
<td>OAREA requests the areas to be written to the unload files. These unload files are defined by the OAREAxxx DD statements.</td>
</tr>
<tr>
<td>23</td>
<td>Area PFPSAMP1 is output using the new DBD parameters. The CI size is changed from 4096 to 8192. The UOW, RAA and IOVF are increased: AREA DD1=PFPSAMP1, SIZE=8192, UOW=(40,10), ROOT(350,100) This change also increases the DASD space required for new area data set allocation.</td>
</tr>
<tr>
<td>24</td>
<td>Area PFPSAMP2 is output using the new DBD parameters. The CI size is changed from 4096 to 8192: AREA DD1=PFPSAMP2, SIZE=8192 This change also increases the DASD space required for new area data set allocation.</td>
</tr>
<tr>
<td>25</td>
<td>Area PFPSAMP3 is output without being changed.</td>
</tr>
<tr>
<td>26</td>
<td>End of UNLOAD control card input.</td>
</tr>
<tr>
<td>27-29</td>
<td>Comment to indicate that the IDCAMS delete/define for the new areas to be reloaded must be performed, with increased space for areas PFPSAMP1 and PFPSAMP2.</td>
</tr>
<tr>
<td>30-33</td>
<td>EXEC and STEPLIB DD statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>34</td>
<td>The DBD from the IMSACB DD statement is used to define the areas being reloaded.</td>
</tr>
<tr>
<td>35-37</td>
<td>Identify input files from UNLOAD.</td>
</tr>
<tr>
<td>38-40</td>
<td>Identify areas to be reloaded.</td>
</tr>
<tr>
<td>41</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>42</td>
<td>Defines DEDB name to be reloaded. SORT=AUTO is specified to determine if an internal sort needs to be performed prior to reloading the areas. If performed, the internal sort places the records into load sequence based on RAP. If you have a license for the BMC Software Fast Path Analyzer/EP product, quick (checksum) pointer validation is performed on reloaded areas by default.</td>
</tr>
<tr>
<td>43</td>
<td>IAREA specifies the unloaded areas to be read. These unload files are defined in the IAREAxxx DD statements.</td>
</tr>
<tr>
<td>44</td>
<td>OAREA specifies the areas to be reloaded. These areas are defined in the PFPSAMPx DD statements.</td>
</tr>
<tr>
<td>45-47</td>
<td>An image copy is taken of the reloaded areas. A new GDG image copy data set is cataloged to DASD.</td>
</tr>
</tbody>
</table>
Use UNLOAD and RELOAD to change compression

The compression routine is changed for three segments based on changes made in the DBD. The compression routine is changed for segment SEGB, removed for segment SEGD, and added for segment SEGC. Minimizing the number of segments to be expanded and compressed will minimize I/O and improve job performance.

**NOTE**
The new ACB with the DBD changes must be generated before executing the RELOAD control statement.

Figure 282  JCL to use UNLOAD and RELOAD to change compression (part 1 of 2)

```
01//PFPUUNLD EXEC PGM=PFPMAIN,REGION=OM
02//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
03// DD DISP=SHR,DSN=IMSVS.DFSMDA
04// DD DISP=SHR,DSN=IMSVS.RESLIB  <-- DFSCMPXO
05// DD DISP=SHR,DSN=BMC.DPK.LOAD  <-- DPIFPRTN
06//$$DPICDS DD DISP=SHR,DSN=PFP.DPK.DPICDS
07//$$DPITBL DD DISP=SHR,DSN=PFP.DPK.DPITBL
08//IMSCB DD DISP=SHR,DSN=IMSVS.ACBLIB
09//NEWACB DD DISP=SHR,DSN=IMSVS.ACBLIB.NEW
10//PFPSAMP1 DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP1
11//PFPSAMP2 DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP2
12//PFPSAMP3 DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP3
13//OUTUNLD DD DISP=(,CATLG,DELETE),UNIT=DASD,SPACE=(CYL,(150,5),RLSE),
14// DSN=PFP.PFPSAMP1.UNLOAD
15//PFPSYSIN DD *
16 UNLOAD DBD=PFPSAMP,
17   IAREA=ALL,
18   EXPAND=(SEGB,SEGD),
19   SDEP_PROCESS=LOGICAL
20 OFILECTL OAREA=ALL,DDNAME=OUTUNLD
21/*
22/** **************************************************
23/** IDCAMS <--- area IDCAMS delete/define
24/** **************************************************
25//PFPPRELD EXEC PGM=PFPMAIN,REGION=OM
26//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
27// DD DISP=SHR,DSN=IMSVS.RESLIB
28// DD DISP=SHR,DSN=IMSVS.DFSMDA
29// DD DISP=SHR,DSN=BMC.DPK.LOAD
30//$$DPICDS DD DISP=SHR,DSN=PFP.DPK.DPICDS
31//$$DPITBL DD DISP=SHR,DSN=PFP.DPK.DPITBL
32//IMSCB DD DISP=SHR,DSN=IMSVS.ACBLIB.NEW
33//IAREA001 DD DISP=OLD,DSN=PFP.PFPSAMP1.UNLOAD
34//PFPSAMP1 DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP1
35//PFPSAMP2 DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP2
36//PFPSAMP3 DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP3
```
Use UNLOAD and RELOAD to change compression

Figure 282  JCL to use UNLOAD and RELOAD to change compression (part 2 of 2)

```
37//PFPSYSIN DD *
38 RELOAD DBD=PFPSAMP,
39     IAREA=PFPSAMP1,
40     OAREA=ALL,
41 COMPRESS=(SEGB,SEGC)
42 IC DSNAME='PFP.ICOPY.&DBD.&AREA(+1)',
43 UNIT=SYSDA,DISP=(NEW,CATLG),
44 SPACE=(CYL,150,5,RLSE)
/*
```

Table 69  Descriptive text for JCL to use UNLOAD and RELOAD to change compression (part 1 of 2)

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-05</td>
<td>EXEC and STEPLIB statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>04</td>
<td>Compression routines specified in NEWACB DD are used on output areas.</td>
</tr>
<tr>
<td>05</td>
<td>Compression routines specified in IMSACB DD are used on input areas.</td>
</tr>
<tr>
<td>06-07</td>
<td>DD statements that define data sets for the BMC Software DATA PACKER®/IMS product. These data sets are required because the segment data must be expanded in order to apply the new compression routines.</td>
</tr>
<tr>
<td>08</td>
<td>The DBD from the IMSACB DD statement is used for input areas.</td>
</tr>
<tr>
<td>10-12</td>
<td>The DBD from the NEWACB DD statement is used for the output unload file.</td>
</tr>
<tr>
<td>13-14</td>
<td>Define input areas to be unloaded.</td>
</tr>
<tr>
<td>15</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>16</td>
<td>Defines DEBD name for UNLOAD processing.</td>
</tr>
<tr>
<td>17</td>
<td>IAREA specifies the input areas for the PFPSAMPx DD statements. All areas are specified since the compression changes affect all the SEGB, SEGC, and SEGD segments.</td>
</tr>
<tr>
<td>18</td>
<td>EXPAND keyword causes SEGB and SEGD statements to be expanded using the compression exit specified for the segment in the IMSACB. The compressed segments in the input are expanded with the compression routines (DPIFPRTN) specified in IMSACB data set. The two changed segments are:</td>
</tr>
<tr>
<td></td>
<td>SEGM NAME=SEGB,BYTES=(100,20),PARENT=SEGA, COMPRTN=(DPIFPRTN,DATA,INIT)</td>
</tr>
<tr>
<td></td>
<td>SEGM NAME=SEGD,BYTES=(100,20),PARENT=SEGB COMPRTN=(DPIFPRTN,DATA,INIT)</td>
</tr>
<tr>
<td>19</td>
<td>Because the DEDB contains SDEPs, SDEP processing must be specified.</td>
</tr>
<tr>
<td>20</td>
<td>One unload file is created. OAREA specifies the area names to be written to the OUTUNLD DD statement.</td>
</tr>
<tr>
<td>21</td>
<td>End of UNLOAD control card input.</td>
</tr>
<tr>
<td>22-24</td>
<td>Comment to indicate that the IDCAMS delete/define for the new areas to be reloaded must be performed.</td>
</tr>
</tbody>
</table>
Increase number of areas and initialize new fields in segments

The number of areas in the database is increased from three to five, and the segment lengths of two segments (SEGC and SEGD) are increased. Segment changes are applied on the reload step for segment SEGC (not compressed) and segment SEGD (compressed.)
Increase number of areas and initialize new fields in segments

**Figure 283   JCL to increase numbers of areas and initialize new fields in segments (part 1 of 2)**

```
01//PFPPUNLD EXEC PGM=PFPPMAIN,REGION=OM
02//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
03// DD DISP=SHR,DSN=IMSVS.RESLIB
04// DD DISP=SHR,DSN=IMSVS.DFSMDA
05//IMSAACB DD DISP=SHR,DSN=IMSVS.ACBLIB
06//NEWACB DD DISP=SHR,DSN=IMSVS.ACBLIB.NEW
07//OAREA001 DD DISP=(,CATLG,DELETE),DSN=PFP.PFPSAMP1.UNLOAD,
08// UNIT=SYSDA,SPACE=(CYL,(150,20),RLSE)
09//OAREA002 DD DISP=(,CATLG,DELETE),DSN=PFP.PFPSAMP2.UNLOAD,
10// UNIT=SYSDA,SPACE=(CYL,(150,20),RLSE)
11//OAREA003 DD DISP=(,CATLG,DELETE),DSN=PFP.PFPSAMP3.UNLOAD,
12// UNIT=SYSDA,SPACE=(CYL,(150,20),RLSE)
13//OAREA004 DD DISP=(,CATLG,DELETE),DSN=PFP.PFPSAMP4.UNLOAD,
14// UNIT=SYSDA,SPACE=(CYL,(150,20),RLSE)
15//OAREA005 DD DISP=(,CATLG,DELETE),DSN=PFP.PFPSAMP5.UNLOAD,
16// UNIT=SYSDA,SPACE=(CYL,(150,20),RLSE)
17//PFPSYSIN DD *
18 UNLOAD DBD=PFPSAMP.
19 IAREA=ALL.
20 OAREA=ALL.
21 INPUT_DSN_MASK='PFP.ICOPY.&DBD.&AREA',
22 SDEP_PROCESS=LOGICAL
23/*
24//*******************************************************************
25//* IDCAMS <--- area IDCAMS delete/defines
26//*******************************************************************
27//PFPPRELD EXEC PGM=PFPPMAIN,REGION=OM
28//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
29// DD DISP=SHR,DSN=IMSVS.RESLIB
30// DD DISP=SHR,DSN=IMSVS.DFSMDA
31// DD DISP=SHR,DSN=BMC.DPK.LOAD
32//$$DPICDS DD DISP=SHR,DSN=PFP.DPK.DPICDS
33//$$DPITBL DD DISP=SHR,DSN=PFP.DPK.DPITBL
34//IMSAACB DD DISP=SHR,DSN=IMSVS.ACBLIB.NEW
35//IAREA001 DD DISP=OLD,DSN=PFP.PFPSAMP1.UNLOAD
36//IAREA002 DD DISP=OLD,DSN=PFP.PFPSAMP2.UNLOAD
37//IAREA003 DD DISP=OLD,DSN=PFP.PFPSAMP3.UNLOAD
38//IAREA004 DD DISP=OLD,DSN=PFP.PFPSAMP4.UNLOAD
39//IAREA005 DD DISP=OLD,DSN=PFP.PFPSAMP5.UNLOAD
40//PFPSYSIN DD *
41 RELOAD DBD=PFPSAMP.
42 IAREA=ALL.
43 OAREA=ALL.
44 OUTPUT_DSN_MASK='BMCPFP.&DBD.&AREA',
45 POINTER_VALIDATION=NONE.
46 EXPAND=(SEGD)
47 OUTPUT SEGMENT=SEGC,
48 FIELDS=(3:8,'000000',11:*)
49 OUTPUT SEGMENT=SEGD,
50 FIELDS=(3:14,
```
Figure 283  JCL to increase numbers of areas and initialize new fields in segments (part 2 of 2)

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>51</td>
<td>17:3P,15:2.</td>
</tr>
<tr>
<td>52</td>
<td>20:*,-5P'0')</td>
</tr>
<tr>
<td>53</td>
<td>IC DSNAMES='PFP.ICOPY.&amp;DBD.&amp;AREA(+1)',</td>
</tr>
<tr>
<td>54</td>
<td>UNIT=SYSDA,DISP=(NEW,CATLG),SPACE=(CYL,150,5,RLSE)</td>
</tr>
</tbody>
</table>

/*

Table 70  Descriptive text for JCL to increase numbers of areas and initialize new fields in segments (part 1 of 2)

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB DD statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05, 21</td>
<td>The DBD from the IMSACB DD statement is used to define the input areas.</td>
</tr>
<tr>
<td>06</td>
<td>The DBD from the NEWACB DD statement is used to define the output unload files.</td>
</tr>
<tr>
<td>07-16</td>
<td>Define output files for unloaded data (one file for each area), including the two new areas.</td>
</tr>
<tr>
<td>17</td>
<td>PFP SYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>18</td>
<td>Defines DEBD name for UNLOAD processing.</td>
</tr>
<tr>
<td>19, 21</td>
<td>IAREA requests that all three areas be processed. These areas are dynamically allocated using the INPUT_DSN_MASK keyword.</td>
</tr>
<tr>
<td>20</td>
<td>OAREA requests that all areas be written to the unload files. These unload areas are defined by the OAREAxxx DD statements, including ones for the two new areas.</td>
</tr>
<tr>
<td>21</td>
<td>INPUT_DSN_MASK defines the input data set names for the database being unloaded. These names specify GDG image copy data sets that are dynamically allocated. A generation number of (0) will be used by default.</td>
</tr>
<tr>
<td>22</td>
<td>Because the DEDB contains SDEPs, SDEP processing must be specified.</td>
</tr>
<tr>
<td>23</td>
<td>End of UNLOAD control card input.</td>
</tr>
<tr>
<td>24-26</td>
<td>Comment to indicate that the IDCAMS delete/define for the new areas to be reloaded must be performed.</td>
</tr>
<tr>
<td>31-33</td>
<td>DD statements that define data sets for the BMC Software DATA PACKER®/IMS product. These data sets are required because segment SEGD is compressed and must be expanded to process the specified changes.</td>
</tr>
<tr>
<td>34</td>
<td>The DBD from the IMSACB DD statement is used to define the areas being reloaded.</td>
</tr>
<tr>
<td>35-39</td>
<td>Define input files from UNLOAD.</td>
</tr>
<tr>
<td>40</td>
<td>PFP SYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>41</td>
<td>Defines DEBD name to be reloaded.</td>
</tr>
<tr>
<td>42</td>
<td>IAREA specifies that all unloaded areas to be read. These unload files are defined by the IAREAxxx DD statements.</td>
</tr>
<tr>
<td>43</td>
<td>OAREA specifies that all areas are to be reloaded.</td>
</tr>
<tr>
<td>44</td>
<td>OUTPUT_DSN_MASK defines a template to be used to create the area names to be reloaded. These data sets must be allocated prior to running this step. In this case, allocation is performed in the IDCAMS step.</td>
</tr>
</tbody>
</table>
Create a test database with a subset of data

A smaller test database is created by using image copy as input to unload and limiting the resulting unload file.

Figure 284  JCL to create a test database with a subset of data (part 1 of 2)

```
01//PFPUNLD  EXEC PGM=PFPMAIN,REGION=0M
02//STEPLIB  DD DISP=SHR,DSN=BMC.PFP.LOAD
03///         DD DISP=SHR,DSN=IMSVS.RESLIB
04//         DD DISP=SHR,DSN=IMSVS.DFSMDA
05//IMSACB   DD DISP=SHR,DSN=IMSVS.ACBLIB
06//NEWACB   DD DISP=SHR,DSN=IMSVS.ACBLIB
07//OUTUNLD  DD DISP=(,CATLG,DELETE),
08//         DSN=PFP.FFPSAMP1.UNLOAD,
```

Table 70  Descriptive text for JCL to increase numbers of areas and initialize new fields in segments (part 2 of 2)

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>Pointer validation and physical validation are not performed. POINTER_VALIDATION=None overrides the default checksum validation.</td>
</tr>
</tbody>
</table>
| 46       | The EXPAND keyword is required because SEGD is a compressed segment that must be expanded to process the specified changes. When specified on a reload statement, the EXPAND keyword functions in the same manner as it does for as for a CHANGE command; SEGD is expanded internally for processing, and is then compressed when written to the area data set. SEGC is not specified since it is not compressed.
| 47-48    | OUTPUT subcommands identify the modified segments to be output. All other database segments are written to the output areas, but are not changed.
| 49       | The segment for SEGC is increased by 6 bytes: SEGM NAME=SEGC,BYTES=(56,13) For SEGC, a new 6-byte character field is inserted after the key. This new field is initialized to zeroes, and the previous data is shifted to a position after the new field. The values that are specified on the FIELDS keyword copies 8 bytes starting in column 3 from the input, 6 bytes of zeroes, followed by the input data starting at column 11 to the end of the input data. |
| 50       | SEGD is written out in the format defined by the FIELDS statement, as follows: Copy the first 14 bytes from the input segment, starting at column 3. |
| 51       | Then, the next 2 fields that are written to the output are swapped from where they were positioned on the input file: Starting at column 17 from the input, copy 3 bytes to the output. Next, starting at column 15 from the input, copy 2 bytes to the output. |
| 52       | Next, starting at column 20 from the input, write out the rest of the segment. Then, append 5 bytes in packed decimal format that are initialized to zero. |
| 53-54    | Image copies are taken of the reloaded database areas. A new GDG image copy data set is cataloged to DASD. |
Create a test database with a subset of data

**Figure 284  JCL to create a test database with a subset of data (part 2 of 2)**

```jcl
09//            UNIT=SYSDA,
10//            SPACE=(CYL,(125,5),RLSE)
11//PFPSYSIN DD *
12 UNLOAD DBD=PFPSAMP,IAREA=ALL,
13 SDEP_PROCESS=LOGICAL,
14 ERROR_THRESHOLD=8,
15 INPUT_DSN_MASK='PFP.ICOPY.&DBD.&AREA(0)'
16 INCLUDE SEGMENT=SEGA,SAMPLE_INTERVAL=5
17 INCLUDE SEGMENT=SEGE,SAMPLE_LIMIT=4
18 INCLUDE SEGMENT=SEGG,SAMPLE_INTERVAL=8,
19 SAMPLE_LIMIT=4
20 OFILECTL OAREA=ALL,DDNAME=OUTUNLD
21/*
22//***************************************************************************
23//* IDCAMS <--- area IDCAMS delete/defines
24***************************************************************************
25//PFPPRELD EXEC PGM=PFPMAIN,REGION=0M
26//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
27// DD DISP=SHR,DSN=IMSVS.RESLIB
28// DD DISP=SHR,DSN=IMSVS.DFSMDA
29//IMSA acb DD DISP=SHR,DSN=IMSVS.ACBLIB
30//IAREA001 DD DISP=SHR,DSN=PFP.PFPSAMP1.UNLOAD
31//PFPSAMP1 DD DISP=OLD,DSN=BMCPFP.TEST.PFPSAMP.PFPSAMP1
32//PFPSAMP2 DD DISP=OLD,DSN=BMCPFP.TEST.PFPSAMP.PFPSAMP2
33//PFPSAMP3 DD DISP=OLD,DSN=BMCPFP.TEST.PFPSAMP.PFPSAMP3
34//PFPSYSIN DD *
35 RELOAD DBD=PFPSAMP,
36 IAREA=PFPSAMP1,
37 OAREA=ALL
/*
```

**Table 71  Descriptive text for JCL to create a test database with a subset of data (part 1 of 2)**

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB DD statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05-06</td>
<td>IMSACB and NEWACB DD statements use the same data set since there is no DBD change.</td>
</tr>
<tr>
<td>07-10</td>
<td>One unload file is created from all three area image copy inputs.</td>
</tr>
<tr>
<td>11</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>12</td>
<td>Defines DEDB name for UNLOAD processing. Data is unloaded from all areas (IAREA=ALL is default).</td>
</tr>
<tr>
<td>13</td>
<td>Because the DEDB contains SDEPs, SDEP processing must be specified.</td>
</tr>
<tr>
<td>14</td>
<td>When unloading the input image copy, the ERROR_THRESHOLD keyword allows for up to 8 segment pointer errors to be bypassed before the unload is terminated. Segments with pointer errors are not processed, and the UNLOAD process continues with the next segment.</td>
</tr>
<tr>
<td>15</td>
<td>The zero generation of the area image copy data sets will be dynamically allocated using the INPUT_DSN_MASK keyword for input to the UNLOAD command.</td>
</tr>
</tbody>
</table>
Unload and reload a DEDB and add (build) associated indexes during reload

The functionality in this scenario requires that you have a license for the Fast Path Indexer/EP product. Two indexes are added to the PFPSAMP database. A separate IX subcommand is specified for each index to be built.

**NOTE**
The new ACB with the DBD changes must be genned before executing the UNLOAD control statement.

---

**Figure 285  JCL to unload and reload a DEDB and add (build) associated indexes during reload (part 1 of 2)**

```plaintext
01//PFPUNLD  EXEC PGM=PFPMAIN,REGION=0M
02//STEPLIB  DD DISP=SHR,DSN=BMC.PFP.LOAD
03//       DD DISP=SHR,DSN=IMSVS.RESLIB
```

---

**Table 71  Descriptive text for JCL to create a test database with a subset of data (part 2 of 2)**

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>Limit the unload file by writing every fifth SEGA (root) segment and its dependent segments to the unload file (approximately 20% of database).</td>
</tr>
<tr>
<td>17</td>
<td>Further limit the unload file by writing out a maximum of four SEGE segments (and all dependents) for each selected SEGA segment.</td>
</tr>
<tr>
<td>18-19</td>
<td>Further limit the unload file by writing out every eighth SEGG segment, up to a maximum of 4 SEGG segments for each selected SEGA segment.</td>
</tr>
<tr>
<td>20</td>
<td>All areas are written to a single output file using the provided DD name.</td>
</tr>
<tr>
<td>21</td>
<td>End of UNLOAD control card input.</td>
</tr>
<tr>
<td>22-24</td>
<td>Comment to indicate that the IDCAMS delete/define for the new areas to be reloaded must be performed.</td>
</tr>
<tr>
<td>29</td>
<td>The DBD from the IMSACB DD statement is used to define the areas being reloaded.</td>
</tr>
<tr>
<td>30</td>
<td>Identifies input unload file.</td>
</tr>
<tr>
<td>31-33</td>
<td>Identify output area data sets.</td>
</tr>
<tr>
<td>34</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>35</td>
<td>Defines the DEDB name to be reloaded. If you have a license for the BMC Software Fast Path Analyzer/EP product, quick (checksum) pointer validation is performed on all areas by default.</td>
</tr>
<tr>
<td>36</td>
<td>IAREA specifies the first area name that correlates to the IAREA001 DD statement, and uses the specified unload file.</td>
</tr>
<tr>
<td>37</td>
<td>OAREA specifies that all areas are to be reloaded.</td>
</tr>
</tbody>
</table>
Unload and reload a DEDB and add (build) associated indexes during reload

Table 72  Descriptive text for JCL to unload and reload a DEDB and add (build) associated indexes during reload (part 1 of 2)

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB DD statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05</td>
<td>The DBD from the IMSACB DD statement is used for input areas.</td>
</tr>
</tbody>
</table>
### Table 72  Descriptive text for JCL to unload and reload a DEDB and add (build) associated indexes during reload (part 2 of 2)

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
</table>
| 06       | The DBD from the NEWACB DD statement is used for output areas. Two indexes are added to the SEGA statement, as indicated by the PFXCHILD and PFXXDFLD statements. The following changes are made in the DBD:  
  ```  
  SEGM NAME=SEGA  
  PFXCHILD NAME=(XSEG1,PFPINDX1),...  
  PFXXDFLD NAME=XFLD,SRCH=SEGAKEY,...  
  PFXCHILD NAME=(XSEG2,PFPINDX2),...  
  PFXXDFLD NAME=XFLD,SRCH=FLD01,STHSEQ=SEGAKEY,...  
  ```  |
| 07-10    | Defines output unload file for all areas. |
| 11       | PFPSYSIN DD for Fast Path/EP control statements. |
| 11       | Defines DEBD name for UNLOAD processing. All areas are unloaded. |
| 13       | Because the DEDB contains SDEPs, SDEP processing must be specified. |
| 14       | Input areas are dynamically allocated using the INPUT_DSN_MASK keyword. |
| 15       | All output areas are written to one unload file, which is specified by the OUTUNLD DD statement. |
| 16       | End of UNLOAD control card input. |
| 17-19    | Comment to indicate that the IDCAMS delete/define for the new areas and indexes to be reloaded must be performed. |
| 20-23    | EXEC and STEPLIB DD statements for Fast Path/EP offline execution. |
| 24-26    | DD statements that define data sets for the BMC Software DATA PACKER®/IMS product. These data sets are required because the index PFPINDEX2 is built using the SRCH field FLD01, which is part of a compressed segment. The compressed segment must be expanded internally before building the index. |
| 27       | The DBD from the IMSACB DD statement is used to define the areas being reloaded. |
| 28       | PFXLIB DD refers to the index registration library. |
| 29       | Identifies input unload file. |
| 30       | Specifies output index allocation and build for PFPINDEX1. |
| 31       | Specifies output index allocation and build for PFPINDEX2. |
| 32       | PFPSYSIN DD for Fast Path/EP control statements. |
| 33       | Defines the DEDB name to be reloaded. IAREA specifies the first area name that correlates to the IAREA001 DD statement, and uses the specified unload file. All areas are reloaded. If you have a license for the BMC Software Fast Path Analyzer/EP product, quick (checksum) pointer validation is performed on all areas by default. |
| 34       | Output areas are dynamically allocated using the substitution variables in the OUTPUT_DSN_MASK for the area data set names. |
| 35-37    | Image copies are taken of the reloaded areas. A new GDG image copy data set is cataloged to DASD. |
| 38       | Build index database PFPINDEX1. |
| 39       | Build index database PFPINDEX2. The SRCH field FLD01 is part of the compressed data for segment SEGA. Segment SEGA is internally expanded to ensure that the index PFPINDEX2 is built with expanded data. |
Unload DEDB and use FABEUR6 and FABEUR7 utilities to change segment data, then reload DEDB

This scenario demonstrates an alternate method for changing segment data by utilizing the Fast Path/EP FABEUR7 (read unload file) and FABEUR6 (create unload file) user application program extensions. As FABEUR7 and FABEUR6 read and write each record, the record is expanded and then recompressed to conserve DASD space in the input and output unload files. For more information, see the Fast Path/EP Series Reference Manual.

Figure 286  JCL to unload DEDB and use FABEUR6 and FABEUR7 utilities to change segment data, then reload DEDB (part 1 of 2)

```
01//PFPUNLD  EXEC PGM=PFPMAIN,REGION=0M
02//STEPLIB  DD DISP=SHR,DSN=BMC.PFP.LOAD
03//         DD DISP=SHR,DSN=IMSVS.RESLIB
04//         DD DISP=SHR,DSN=IMSVS.DFSMDA
05//IMSACB  DD DISP=SHR,DSN=IMSVS.ACBLIB
06//NEWACB  DD DISP=SHR,DSN=IMSVS.ACBLIB
07//PFPSAMP1 DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP1
08//PFPSAMP2 DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP2
09//PFPSAMP3 DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP3
10//OUTUNLD DD DSN=PFP.PFPSAMP1.UNLOAD,DISP=(,CATLG,DELETE),
           UNIT=SYSDA,SPACE=(CYL,(200,15),RLSE)
12//PFPSYSIN DD *
13 UNLOAD DBD=PFPSAMP,IAREA=ALL,
14 SDEP_PROCESS=LOGICAL,
15 OFILECTL OAREA=ALL,DDNAME=OUTUNLD
16/*
17//*******************************************************************************
18//FABEUR67 EXEC PGM=FAB67,REGION=0M
19//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
20// DD DISP=SHR,DSN=IMSVS.RESLIB
21// DD DISP=SHR,DSN=YOUR.PGMLOAD
22// DD DISP=SHR,DSN=BMC.DPK.LOAD
23//$$DPICDS DD DISP=SHR,DSN=PFP.DPK.DPICDS
24//$$DPICTBL DD DISP=SHR,DSN=PFP.DPK.DPICTBL
25//ACBLIB DD DISP=SHR,DSN=IMSVS.ACBLIB
26//UR7DATA DD DISP=SHR,DSN=PFP.PFPSAMP1.UNLOAD
27//SYSPRINT DD SYSOUT=* 
28//UR7RPT DD SYSOUT=* 
29//UR7CTL DD *
30 DLICOMP=EXP
31/*
32//DURD0010 DD DSN=PFP.PFPSAMP1.UNLOAD.NEW,
33// Disp=(,CATLG,DELETE),UNIT=SYSDA,
34 // SPACE=(CYL,(200,15),RLSE)
35//UR6RPT DD SYSOUT=*,DCB=(LRECL=121,RECFM=FBA)
36//UR6FCTL DD *
37 FILECTL=1,ALL
```
Unload DEDB and use FABEUR6 and FABEUR7 utilities to change segment data, then reload DEDB

Figure 286  JCL to unload DEDB and use FABEUR6 and FABEUR7 utilities to change segment data, then reload DEDB (part 2 of 2)

Table 73  Descriptive text for JCL to unload DEDB and use FABEUR6 and FABEUR7 utilities to change segment data, then reload DEDB (part 1 of 2)

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB DD statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05-06</td>
<td>The data set name from IMSACB and NEWACB is the same because the DBD definition has not changed.</td>
</tr>
<tr>
<td>07-09</td>
<td>Define all input area data sets.</td>
</tr>
<tr>
<td>10-11</td>
<td>The unload output for all areas is written to the DD name OUTUNLD.</td>
</tr>
<tr>
<td>12</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>13</td>
<td>Defines DEBD name for UNLOAD processing. All areas are unloaded.</td>
</tr>
<tr>
<td>14</td>
<td>Because the DEDB contains SDEPs, SDEP processing must be specified.</td>
</tr>
<tr>
<td>15</td>
<td>All output areas are written to one unload file, which is specified by the OUTUNLD DD statement.</td>
</tr>
<tr>
<td>16,17</td>
<td>End of UNLOAD control card input. Break line as visual indicator of start of user-written program (FAB67).</td>
</tr>
<tr>
<td>18</td>
<td>User-written program (FAB67) that is executed to make changes to the data in the unloaded segments. The program utilizes the FABEUR7 extension to read the HDUNLOAD-format unload file and the FABEUR6 extension to write a new HDUNLOAD-format file with the modified segments. The program will make the following changes: All Julian dates in the segments are converted from 5-byte character fields (YYDDD) to 5-byte packed fields with 4-digit years (00YYYYDDDF). Also, several fields are initialized to zeroes depending on specific criteria.</td>
</tr>
<tr>
<td>19-20</td>
<td>STEPLIB DD statements for user program and Fast Path/EP offline execution.</td>
</tr>
</tbody>
</table>
Define different space allocations for reload output areas

This technique can also be used on a CHANGE or INITIALIZE command.
Define different space allocations for reload output areas

**Figure 287  JCL to define different space allocations for reload output areas**

```jcl
01//PFP EXEC PGM=PFPMAIN,REGION=0M
02//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD,
03// DD DISP=SHR,DSN=IMSVS.RESLIB
04// DD DISP=SHR,DSN=IMSVS.DFSMDA
05//IMSACB DD DISP=SHR,DSN=IMSVS.ACBLIB
07//IAREA001 DD DISP=SHR,DSN=PFP.PFPSAMP1.UNLOAD
08//IAREA002 DD DISP=SHR,DSN=PFP.PFPSAMP2.UNLOAD
09//PFPSYSIN DD *
10 RELOAD DBD=PFPSAMP,IAREA=(PFPSAMP1,PFPSAMP2),
11 OAREA=(PFPSAMP1,PFPSAMP2)
12 ALLOCATE OAREA=PFPSAMP1,
13 ACTUATE=(DELETE,EROPT=IGNORE),
14 SPACE=(CYL,200),VOLSER=(vvvvvv)
15 ALLOCATE OAREA=PFPSAMP2,
16 ACTUATE=(DELETE,EROPT=IGNORE),
17 SPACE=(CYL,300),VOLSER=(vvvvvv)
/*
```

**Table 74  Descriptive text for JCL to define different space allocations for reload output areas**

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB DD statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>06</td>
<td>The DBD from the IMSACB DD statement is used for output areas.</td>
</tr>
<tr>
<td>07-08</td>
<td>Input area data sets.</td>
</tr>
<tr>
<td>09</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>10</td>
<td>Defines the DEDB name to be reloaded. IAREA specifies the unloaded areas to be read. These unload files are defined in the IAREAxxx DD statements.</td>
</tr>
<tr>
<td>11</td>
<td>OAREA specifies the areas to be reloaded. The two areas are dynamically allocated from their dynamic allocation members in the IMSVS.DFSMDA data set in the STEPLIB concatenation.</td>
</tr>
<tr>
<td>12</td>
<td>Area PFPSAMP1 is allocated. The OAREA keyword is required to identify the area name.</td>
</tr>
<tr>
<td>15</td>
<td>Area PFPSAMP2 is allocated. The OAREA keyword is required to identify the area name.</td>
</tr>
<tr>
<td>12-13</td>
<td>The DELETE parameter that is specified on the ACTUATE keyword will cause the current VSAM cluster to be deleted. If the VSAM cluster did not previously exist, the error condition would be bypassed by the IGNORE option that is specified on the EROPT parameter.</td>
</tr>
<tr>
<td>14</td>
<td>The new VSAM clusters (one for each output area) will be defined according to the unique attributes that are specified for each area by the SPACE and VOLSER keywords.</td>
</tr>
</tbody>
</table>
Offline EXTRACT command

The scenarios in this section show how to use the offline EXTRACT command in conjunction with key related keywords and subcommands. Collectively, these examples demonstrate all options that are available for customizing the output of a data extract. To exemplify segment selection/exclusion, the scenarios in this section use the segment hierarchy shown on page 361.

Table 75  EXTRACT scenarios

<table>
<thead>
<tr>
<th>Primary command/scenario task</th>
<th>Subcommand/keyword</th>
<th>Concept/process</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXTRACT Extract and Output Only Selected Segments</td>
<td>EXTRACT_FORMAT=EXTRACT EXPAND EXCLUDE SEGMENT with WHERE OFILECTL with OAREA OUTPUT SEGMENT</td>
<td>■ conditional requirement to expand data  ■ specify extract format  ■ conditionally exclude segments  ■ relationship between excluded data and output data  ■ define output files</td>
<td>410</td>
</tr>
<tr>
<td>EXTRACT Generate a Subset of Area Data</td>
<td>EXTRACT_FORMAT=HDUNLOAD INCLUDE SEGMENT SAMPLE_LIMIT SAMPLE_INTERVAL OFILECTL</td>
<td>■ create output file in HDUNLOAD format  ■ include sampling of root segments of interval and limit criteria  ■ define and dynamically allocate output files</td>
<td>411</td>
</tr>
<tr>
<td>EXTRACT Extract a Selected Segment and Sort Data</td>
<td>EXTRACT_FORMAT=EXTRACT SORT EXPAND OUTPUT SEGMENT with FIELDS</td>
<td>■ sort output data  ■ dynamic allocation of output files  ■ expand compressed segments</td>
<td>412</td>
</tr>
<tr>
<td>EXTRACT Modify Segment Layout during Extract</td>
<td>INPUT_DSN_MASK EXTRACT_FORMAT=HDUNLOAD EXPAND OFILECTL OUTPUT SEGMENT with FIELDS</td>
<td>■ dynamic allocation of input areas  ■ create output file in HDUNLOAD format  ■ expand compressed data  ■ define and dynamically allocate output files  ■ manipulate output fields in specified segments</td>
<td>413</td>
</tr>
<tr>
<td>EXTRACT Create a Customized Extract File Format</td>
<td>EXTRACT_FORMAT=USER_RECORD with BREAK / BEFORE USER_RECORD with BREAK / AFTER FIELDS SEGMENT_RECORD_PREFIX</td>
<td>■ create customized format for output file  ■ define multiple header records  ■ define trailer record  ■ specify prefix for segment record</td>
<td>415</td>
</tr>
</tbody>
</table>
Extract and output only selected segments

This scenario extracts and writes only two segment types to the output extract file.

Figure 288  JCL to extract and output only selected segments

```
01//PFPMAIN EXEC PGM=PFPMAIN,REGION=0M
02//STEPLIB DD DISP=SHR,DSN=BMC.PFP.PFP.LOAD
03// DD DISP=SHR,DSN=IMSVS.RESLIB
04// DD DISP=SHR,DSN=IMSVS.DFSMDA
05// DD DISP=SHR,DSN=BMC.DPK.LOAD
06//$DPICDS DD DISP=SHR,DSN=PFP.DPK.DPICDS
07//$DPITBL DD DISP=SHR,DSN=PFP.DPK.DPITBL
08//IMSACB DD DISP=SHR,DSN=IMSVS.ACLLIB
09//PFPSAMP1 DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP1
10//PFPSAMP2 DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP2
11//PFPSAMP3 DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP3
12//EXTFILE1 DD DSN=BMC.PFP.EXTFILE1,
13// DISP=(,CATLG,DELETE),
14// UNIT=DISK,
15// SPACE=(CYL,(200,8),RLSE)
16//EXTFILE2 DD DSN=BMC.PFP.EXTFILE2,
17// DISP=(,CATLG,DELETE),
18// UNIT=DISK,
19// SPACE=(CYL,(200,8),RLSE)
20//PFPSYSIN DD *
21 EXTRACT DBD=PFPSAMP,IAREA=ALL,
22 EXTRACT_FORMAT=EXTRACT,
23 EXPAND=YES
24 EXCLUDE SEGMENT=SEGB,
25 WHERE=(SEGA.SEGAKEY LT C'19980101'
26 OUTPUT SEGMENT=SEGA
27 OUTPUT SEGMENT=SEGB
28 OFILECTL OAREA=(PFPSAMP1,PFPSAMP2),DDNAME=EXTFILE1
29 OFILECTL OAREA=PFPSAMP3,DDNAME=EXTFILE2
/*
```

Table 76  Descriptive text for JCL to extract and output only selected segments (part 1 of 2)

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05-07</td>
<td>DD statements that define data sets for the BMC DATA PACKER/IMS product. These data sets are required because expansion of data was requested by the EXPAND=YES keyword, and because of data manipulation for compressed fields.</td>
</tr>
<tr>
<td>08</td>
<td>IMSACB DD statement used for database processing.</td>
</tr>
<tr>
<td>09-11</td>
<td>Input area data sets.</td>
</tr>
<tr>
<td>12-15</td>
<td>The extracted output for areas PFPSAMP1 and PFPSAMP2 is written to DD name EXTFILE1.</td>
</tr>
</tbody>
</table>
Generate a subset of area data

This scenario creates an unload file containing a subset of the area data. The unload file can then be input into reload to create a smaller test database.

**Figure 289  JCL to generate a subset of area data**

```
01//PFPMAIN  EXEC PGM=PFPMAIN,REGION=OM
02//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
03//  DD DISP=SHR,DSN=IMSVS.RESLIB
04//  DD DISP=SHR,DSN=IMSVS.DFSMDA
05//IMSA CB DD DISP=SHR,DSN=IMSVS.ACBLIB
06//PPFSYSIN DD *
07 EXTRACT DBD=PFPSAMP,
08   EXTRACT_FORMAT=HDUNLOAD
09 INCLUDE SEGMENT=SEGA,SAMPLE_INTERVAL=5
10 INCLUDE SEGMENT=SEGE,SAMPLE_LIMIT=4
11 INCLUDE SEGMENT=SEGG,
12   SAMPLE_INTERVAL=8,SAMPLE_LIMIT=10
13 OFILECTL OAREA=ALL,
14 DSNAME='PFP.&DBD.UNLOAD',
15   DISP=(,CATLG,DELETE),
16 DATACLASS=dataclass
/*
```
Table 77   Descriptive text for JCL to generate a subset of area data

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05</td>
<td>IMSACB DD statement used for database processing.</td>
</tr>
<tr>
<td>06</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>07</td>
<td>Defines DEDB name for EXTRACT process. Data is extracted from all areas (IAREA=ALL is default). The area statements are allocated from their dynamic allocation members in the IMSVS.DFSMDA data set in the STEPLIB concatenation.</td>
</tr>
<tr>
<td>08</td>
<td>Data is extracted in HDUNLOAD format, which is suitable for use as input to RELOAD.</td>
</tr>
<tr>
<td>09</td>
<td>Limit the selection by writing every fifth SEGA (root) segment and its dependent segments to the unload file. (approximately 20% of database).</td>
</tr>
<tr>
<td>10</td>
<td>Further limit the selection by writing out a maximum of four SEGE segments (and all dependents) for each selected SEGA segment.</td>
</tr>
<tr>
<td>11-12</td>
<td>Further limit the selection by writing out every eighth SEGG segment, up to a maximum of 10 SEGG segments for each selected SEGA segment.</td>
</tr>
<tr>
<td>13-15</td>
<td>All areas are written to a single output file using the provided data set name.</td>
</tr>
<tr>
<td>16</td>
<td>A data class is specified for this SMS-managed data set.</td>
</tr>
</tbody>
</table>

Extract a selected segment and sort data

Only segment SEGD is extracted and written, with some data modification. SEGD is also sorted in RAP sequence, using the root SEGA key contained in the prefix of the output extract file.

Figure 290   JCL to extract a selected segment and sort data (part 1 of 2)

```plaintext
01//PFPMAIN EXEC PGM=PFPMAIN,REGION=0M  
02//STEPLIB DD DISP=SHR,DSN=BMCPFP.LOAD  
03// DD DISP=SHR,DSN=IMSVS.RESLIB  
04// DD DISP=SHR,DSN=IMSVS.DFSMDA  
05// DD DISP=SHR,DSN=BMC.DPK.LOAD  
06//$DPICDS DD DISP=SHR,DSN=PFP.DPK.DPICDS  
07//$DPITBL DD DISP=SHR,DSN=PFP.DPK.DPITBL  
08//IMSACB DD DISP=SHR,DSN=IMSVS.ACBLIB  
09//OAREA001 DD DSN=PFP.PFPSAMP1.SORTED.EXTRACT,  
10// UNIT=SYSDA,DISP=(),CATLG,SPACE=(CYL,(200,20),RLSE)  
11//OAREA003 DD DSN=PFP.PFPSAMP3.SORTED.EXTRACT,  
12// UNIT=SYSDA,DISP=(),CATLG,SPACE=(CYL,(200,20),RLSE)  
13//PFPSYSIN DD *  
14 EXTRACT DBD=PFPSAMP,IAREA=(PFPSAMP1,PFPSAMP3),  
15 SORT=YES,  
16 EXPAND=YES  
17 OUTPUT SEGMENT=SEGD,  
18 FIELDS=(SEGMENT_CKEY,  
```
Modify segment layout during extract

This scenario uses an image copy as input to the extract. Three segments are modified and output in preparation for a future DBD change using reload.

Figure 290  JCL to extract a selected segment and sort data (part 2 of 2)

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05-07</td>
<td>DD statements that define data sets for the BMC DATA PACKER/IMS product. These data sets are required because expansion of data was requested by the EXPAND=YES keyword, and because of data manipulation for compressed fields.</td>
</tr>
<tr>
<td>08</td>
<td>IMSACB DD statement used for database processing.</td>
</tr>
<tr>
<td>09-10</td>
<td>Extract output file for area PFPSAMP1.</td>
</tr>
<tr>
<td>11-12</td>
<td>Extract output file for area PFPSAMP3.</td>
</tr>
<tr>
<td>13</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>14</td>
<td>Defines DEDB name for EXTRACT process. The two input areas are dynamically allocated from their dynamic allocation members in the IMSVS.DFSMDA data set in the STEPLIB concatenation. Data is extracted in the standard default format (EXTRACT_FORMAT=EXTRACT). The format of the records will use the layout specified in Appendix B in the Fast Path/EP Series Reference Manual.</td>
</tr>
<tr>
<td>15</td>
<td>The extracted output file is sorted and written in root key sequence.</td>
</tr>
<tr>
<td>16</td>
<td>Compressed segments are expanded.</td>
</tr>
<tr>
<td>17</td>
<td>Only segment SEGD is written to the output file.</td>
</tr>
<tr>
<td>18</td>
<td>The contents of SEGD is modified in the output file. The built-in variable SEGMENT_CKEY specifies that the first field will contain the concatenated key of SEGD (value includes the concatenated key length prefix).</td>
</tr>
<tr>
<td>19</td>
<td>The next value is FLD02 from segment SEGA.</td>
</tr>
<tr>
<td>20</td>
<td>The next value is derived by executing the expression: The 4-byte packed decimal field in column 38 in segment SEGB is added to the 4-byte packed decimal field in column 44 in segment SEGB.</td>
</tr>
<tr>
<td>21</td>
<td>Data is written to the extracted output file starting from input segment column 11 of SEGD to the end of the input data.</td>
</tr>
</tbody>
</table>

Modify segment layout during extract

Table 78  Descriptive text for JCL to extract a selected segment and sort data

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05-07</td>
<td>DD statements that define data sets for the BMC DATA PACKER/IMS product. These data sets are required because expansion of data was requested by the EXPAND=YES keyword, and because of data manipulation for compressed fields.</td>
</tr>
<tr>
<td>08</td>
<td>IMSACB DD statement used for database processing.</td>
</tr>
<tr>
<td>09-10</td>
<td>Extract output file for area PFPSAMP1.</td>
</tr>
<tr>
<td>11-12</td>
<td>Extract output file for area PFPSAMP3.</td>
</tr>
<tr>
<td>13</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>14</td>
<td>Defines DEDB name for EXTRACT process. The two input areas are dynamically allocated from their dynamic allocation members in the IMSVS.DFSMDA data set in the STEPLIB concatenation. Data is extracted in the standard default format (EXTRACT_FORMAT=EXTRACT). The format of the records will use the layout specified in Appendix B in the Fast Path/EP Series Reference Manual.</td>
</tr>
<tr>
<td>15</td>
<td>The extracted output file is sorted and written in root key sequence.</td>
</tr>
<tr>
<td>16</td>
<td>Compressed segments are expanded.</td>
</tr>
<tr>
<td>17</td>
<td>Only segment SEGD is written to the output file.</td>
</tr>
<tr>
<td>18</td>
<td>The contents of SEGD is modified in the output file. The built-in variable SEGMENT_CKEY specifies that the first field will contain the concatenated key of SEGD (value includes the concatenated key length prefix).</td>
</tr>
<tr>
<td>19</td>
<td>The next value is FLD02 from segment SEGA.</td>
</tr>
<tr>
<td>20</td>
<td>The next value is derived by executing the expression: The 4-byte packed decimal field in column 38 in segment SEGB is added to the 4-byte packed decimal field in column 44 in segment SEGB.</td>
</tr>
<tr>
<td>21</td>
<td>Data is written to the extracted output file starting from input segment column 11 of SEGD to the end of the input data.</td>
</tr>
</tbody>
</table>
Figure 291  JCL to modify segment layout during extract (part 2 of 2)

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>04-07</td>
<td>DD statements that define data sets for the BMC Software DATA PACKER®/IMS product. These data sets are required because expansion of the data was requested by the EXPAND=YES keyword, and because of data manipulation for compressed fields.</td>
</tr>
<tr>
<td>08</td>
<td>The DBD from the IMSACB DD statement is used for output areas.</td>
</tr>
<tr>
<td>09</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>10</td>
<td>Defines DEDB name for EXTRACT process.</td>
</tr>
<tr>
<td>11</td>
<td>Input for all areas is the zero generation image copy, which is dynamically allocated using the data set name specified with the INPUT_DSN_MASK keyword.</td>
</tr>
<tr>
<td>12</td>
<td>Data is extracted in HDUNLOAD format, which is suitable for use as input to RELOAD.</td>
</tr>
<tr>
<td>13</td>
<td>Compressed data is expanded.</td>
</tr>
<tr>
<td>14-17</td>
<td>All areas are written to a single output file using the provided data set name.</td>
</tr>
<tr>
<td>18-19</td>
<td>Modify data in SDEP segments, by increasing the output length by 12 bytes.</td>
</tr>
<tr>
<td>19</td>
<td>Copy the original input segment in its entirety, starting at column 3. Then, append a 4-byte packed field (initialized to zero), and an 8-byte character field (initialized to blanks)</td>
</tr>
<tr>
<td>20-21</td>
<td>Modify data in SEGC segments by increasing the output length by 3 bytes. Copy the first 12 bytes (columns 3-14) unchanged, then insert a new 3-byte packed field (initialized to zero), followed by the remainder of the input segment from column 15 to the end.</td>
</tr>
</tbody>
</table>

Table 79  Descriptive text for JCL to modify segment layout during extract (part 1 of 2)

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05-07</td>
<td>DD statements that define data sets for the BMC Software DATA PACKER®/IMS product. These data sets are required because expansion of the data was requested by the EXPAND=YES keyword, and because of data manipulation for compressed fields.</td>
</tr>
<tr>
<td>08</td>
<td>The DBD from the IMSACB DD statement is used for output areas.</td>
</tr>
<tr>
<td>09</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>10</td>
<td>Defines DEDB name for EXTRACT process.</td>
</tr>
<tr>
<td>11</td>
<td>Input for all areas is the zero generation image copy, which is dynamically allocated using the data set name specified with the INPUT_DSN_MASK keyword.</td>
</tr>
<tr>
<td>12</td>
<td>Data is extracted in HDUNLOAD format, which is suitable for use as input to RELOAD.</td>
</tr>
<tr>
<td>13</td>
<td>Compressed data is expanded.</td>
</tr>
<tr>
<td>14-17</td>
<td>All areas are written to a single output file using the provided data set name.</td>
</tr>
<tr>
<td>18-19</td>
<td>Modify data in SDEP segments, by increasing the output length by 12 bytes.</td>
</tr>
<tr>
<td>19</td>
<td>Copy the original input segment in its entirety, starting at column 3. Then, append a 4-byte packed field (initialized to zero), and an 8-byte character field (initialized to blanks)</td>
</tr>
<tr>
<td>20-21</td>
<td>Modify data in SEGC segments by increasing the output length by 3 bytes. Copy the first 12 bytes (columns 3-14) unchanged, then insert a new 3-byte packed field (initialized to zero), followed by the remainder of the input segment from column 15 to the end.</td>
</tr>
</tbody>
</table>
Create a customized extract file format

Database PFPSAMP contains three dynamically allocated areas. The
EXTRACT_FORMAT=USER option is specified to enable a customized extract file to
be produced in the format that is required by an existing application program.
Because no EXCLUDE or OUTPUT subcommands are specified in the scenario, all
segments in all areas of PFPSAMP are output. The table on page 417 lists some
examples of the output data.

Table 79  Descriptive text for JCL to modify segment layout during extract (part 2 of 2)

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>22-23</td>
<td>Modify data in SEGG segments, but the first output field is the segment key.</td>
</tr>
<tr>
<td>24</td>
<td>The next two fields that are written to the output are swapped from where they were positioned on the input file: Starting at column 8 from the input, copy 9 bytes to the output. Next, starting at column 5 from the input, copy 3 bytes to the output.</td>
</tr>
<tr>
<td>25</td>
<td>Because the output position in the FIELDS list is the same as the input position of the field being tested, the effect is to conditionally modify the field. If the contents of the two-byte field in column 18 from the input segment is equal to binary zeroes, then write two blanks to the output file. Otherwise, write the input field unchanged.</td>
</tr>
<tr>
<td>26</td>
<td>Write the remainder of the segment unchanged.</td>
</tr>
</tbody>
</table>

Create a customized extract file format

Figure 292  JCL to create a customized extract file format (part 1 of 2)

```plaintext
01//PFPMAIN EXEC PGM=PFPMAIN,REGION=OM
02//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
03// DD DISP=SHR,DSN=IMSVS.RESLIB
04// DD DISP=SHR,DSN=IMSVS.DFSMDA
05//IMSACB DD DISP=SHR,DSN=IMSVS.ACBLIB
06//OAREA001 DD DSN=BMC.PFP.USEROUT1,DISP=(,CATLG,DELETE),
07 // UNIT=SYSDA,SPACE=(CYL,(5,1),RLSE)
08//OAREA002 DD DSN=BMC.PFP.USEROUT2,DISP=(,CATLG,DELETE),
09 // UNIT=SYSDA,SPACE=(CYL,(5,1),RLSE)
10//OAREA003 DD DSN=BMC.PFP.USEROUT3,DISP=(,CATLG,DELETE),
11 // UNIT=SYSDA,SPACE=(CYL,(5,1),RLSE)
12//PFPSYSIN DD *
13 EXTRACT DBD=PFPSAMP,IAREA=ALL,EXTRACT_FORMAT=USER,
14   SEGMENT_RECORD_PREFIX=(X'0003',
15     :2X :=SEGMENT_CODE,
16     SEGMENT_NAME)
17 USER_RECORD BREAK=(AREA,BEFORE),
18 FIELDS=(X'0000',
19 AREA_NAME,
20 :2X :=AREA_NUMBER,
21 :'YYYY%0000DD%0000MM%0000SS%':=SYSDATETIME())
22 USER_RECORD BREAK=(AREA,BEFORE),
23 FIELDS=(X'00001',
```

Appendix C  Sample command scenarios 415
Create a customized extract file format

Figure 292  JCL to create a customized extract file format (part 2 of 2)

```plaintext
24 DBD_NAME,
25 DBD_areas, DBD_segments)
26 USER_RECORD BREAK=(AREA, AFTER),
27 FIELDS=(X'0004',
28 X'0000',
29 SEGMENT_COUNT(1),
30 SEGMENT_COUNT(),
31 F'0')
/*
```

Table 80  Descriptive text for JCL to create a customized extract file format (part 1 of 2)

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05</td>
<td>IMSACB DD statement used for database processing.</td>
</tr>
<tr>
<td>06-07</td>
<td>Extract output file for area PFPSAMP1.</td>
</tr>
<tr>
<td>08-09</td>
<td>Extract output file for area PFPSAMP2.</td>
</tr>
<tr>
<td>10-11</td>
<td>Extract output file for area PFPSAMP3.</td>
</tr>
<tr>
<td>12</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>13</td>
<td>Defines DEDB name for EXTRACT process. All areas are input. Data is extracted from all segments for all areas, using EXTRACT_FORMAT=USER. This format enables use of any customized format that is needed. The areas are dynamically allocated from their dynamic allocation members in the IMSVS.DFSMDA data set in the STEPLIB concatenation.</td>
</tr>
<tr>
<td>14</td>
<td>All segments are written to the output file. Segment records will have a common prefix that will be written first, before each segment data (segment data includes LL field for variable length). The contents of the common prefix is specified by using the SEGMENT_RECORD_PREFIX keyword. The first prefix field for the segment record is X'0003'.</td>
</tr>
<tr>
<td>15</td>
<td>The next field of the segment prefix is performing an assignment (:=) of the one-byte built-in variable of SEGMENT_CODE into a two-byte assignment variable (:2X)</td>
</tr>
<tr>
<td>16</td>
<td>The last field of the segment prefix is an eight-byte built-in variable for SEGMENT_NAME.</td>
</tr>
<tr>
<td>17, 22</td>
<td>For each area, two header records are produced which will be written first (AREA,BEFORE) in the extract output file.</td>
</tr>
<tr>
<td>17</td>
<td>The first header record is created by specifying the USER_RECORD subcommand. The BREAK keyword signifies that this record is to be written before the rest of the area information is written (AREA,BEFORE).</td>
</tr>
<tr>
<td>18</td>
<td>The FIELDS keyword specifies the content of the header record, as follows: record identifier of X'0000'</td>
</tr>
<tr>
<td>19</td>
<td>followed by an 8-byte built-in variable of AREA_NAME</td>
</tr>
<tr>
<td>20</td>
<td>Next is the assignment (:=) of 1-byte built-in variable AREA_NUMBER (2-byte built-in variable for IMS version 8.1 and above) into a two-byte assignment variable (:2X)</td>
</tr>
<tr>
<td>21</td>
<td>Next is a date field with an assignment (:=) of 12-byte calendar date, of the built-in function SYSDATE() into the specified string literal. The literal indicates the time in military time by specifying hours with %24%.</td>
</tr>
</tbody>
</table>
Table 80  
Descriptive text for JCL to create a customized extract file format (part 2 of 2)

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>The second header record is created by specifying the USER_RECORD subcommand. The BREAK keyword signifies that this record is to be written before the rest of the area information is written.</td>
</tr>
<tr>
<td>23</td>
<td>FIELDS keyword specifies the content of the second header record, as follows: record identifier X’0001’ followed by an 8-byte built-in variable for DBD_NAME</td>
</tr>
<tr>
<td>24</td>
<td>Next are the built-in variables for the number of DBD AREAS and the number of segment types for DBD SEGMENTS, each specified as two fixed bytes.</td>
</tr>
<tr>
<td>25</td>
<td>An area trailer record follows all segment records from the area. The area trailer record is created by specifying the USER_RECORD subcommand. The BREAK keyword signifies that this record is to be written after the rest of the area information is written (AREA,AFTER).</td>
</tr>
<tr>
<td>26</td>
<td>FIELDS keyword specifies the content of the trailer record, as follows: record identifier X’0004’ followed by two bytes of zeroes (X’0000’)</td>
</tr>
<tr>
<td>27</td>
<td>Next is the 4-byte built-in function of SEGMENT_COUNT (1), giving the total number of root segments to be processed. The root segment is indicated by the segment code ‘1’ (default with no parm given).</td>
</tr>
<tr>
<td>28</td>
<td>Next is the 4-byte built-in function of SEGMENT_COUNT(), giving the total of all segments processed.</td>
</tr>
<tr>
<td>29</td>
<td>Next is 4 bytes of zeroes as specified by (F’0’) using the default length 4 for the ‘F’ data type.</td>
</tr>
</tbody>
</table>

Table 81 provides some examples of the data output from the JCL that is executed to create a customized extract file format.

Table 81  
Data output from JCL to create a customized extract file format (part 1 of 2)

<table>
<thead>
<tr>
<th>Format</th>
<th>Data output</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>..PFPSAMP1..19990520175735 00DCDECDDF00FFFFFFFFFFFF 00767214710119990520175735</td>
<td>record=x’0000’ arename=PFPSAMP1 area number=x’0001’ date-time=1999/05/20 17:57:35</td>
</tr>
<tr>
<td>hex</td>
<td>..PFPSAMP1.... 00DCDECDDF0000</td>
<td>record=x’0001’ dbdname=PFPSAMP number of areas=x’0003’ number of segments=x’0006’</td>
</tr>
<tr>
<td>char</td>
<td>....SEGA ..4033422007439 0000ECCC444400FFFFFFFFFFFF 0301257100000F4033422007439</td>
<td>record type=x’0003’ segment code=x’0001’ segment name=SEGA LL of segment data=x’000F’ followed by segment data</td>
</tr>
<tr>
<td>char</td>
<td>....SEGB ...... 0000ECCC444400000964 03022572000007801C0</td>
<td>record type=x’0003’ segment code=x’0002’ segment name=SEGB LL of segment data=x’0007’ followed by segment data</td>
</tr>
</tbody>
</table>
The scenarios in this section show how to use the INITIALIZE command in conjunction with key related keywords and subcommands.

Table 82  INITIALIZE scenarios

<table>
<thead>
<tr>
<th>Primary command/scenario task</th>
<th>Subcommand/keyword</th>
<th>Concept/process</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INITIALIZE</td>
<td>IC</td>
<td>initialize areas with basic INITIALIZE command set</td>
<td>419</td>
</tr>
<tr>
<td>Initialize All Areas for a Database</td>
<td>IC with STACK_NAME</td>
<td>create stacked image copy for initialized areas</td>
<td></td>
</tr>
<tr>
<td>INITIALIZE</td>
<td>ALLOCATE</td>
<td>reuse area defined by IAREA on primary command</td>
<td>419</td>
</tr>
<tr>
<td>Initialize and Overwrite an Existing Area</td>
<td>ACTUATE CONFIGURE_AREA SPACE / VOLSER IDCAMS_OPTION</td>
<td>define new VSAM cluster if area does not exist</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>specify size of three separate extents</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>define space allocation unit</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>specify optional IDCAMS definition values</td>
<td></td>
</tr>
</tbody>
</table>

Table 81  Data output from JCL to create a customized extract file format (part 2 of 2)

<table>
<thead>
<tr>
<th>Format</th>
<th>Data output</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>....SEGD ..... PREVIOUS BILLED AMOUNT 66.36</td>
<td>record type=x’0003’ segment code=x’0004’ segment name=SEGD LL of segment data=x’003A’ followed by segment data</td>
</tr>
<tr>
<td>hex</td>
<td>0000ECCC44440301044DDCECDEE4CCDDCC4CDDEDE4FF4FF</td>
<td>record type=x’0004’ half word zeroes=x’0000’ count of segment 01=x’0000015B’ total segment count=x’00001FFD’ word of zeroes=x’00000000’</td>
</tr>
<tr>
<td></td>
<td>0304257400000A0FF007955964202933540146453066B36</td>
<td></td>
</tr>
</tbody>
</table>

INITIALIZE command

Table 81  Data output from JCL to create a customized extract file format (part 2 of 2)

<table>
<thead>
<tr>
<th>Format</th>
<th>Data output</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>....SEGD ..... PREVIOUS BILLED AMOUNT 66.36</td>
<td>record type=x’0003’ segment code=x’0004’ segment name=SEGD LL of segment data=x’003A’ followed by segment data</td>
</tr>
<tr>
<td>hex</td>
<td>0000ECCC44440301044DDCECDEE4CCDDCC4CDDEDE4FF4FF</td>
<td>record type=x’0004’ half word zeroes=x’0000’ count of segment 01=x’0000015B’ total segment count=x’00001FFD’ word of zeroes=x’00000000’</td>
</tr>
<tr>
<td></td>
<td>0304257400000A0FF007955964202933540146453066B36</td>
<td></td>
</tr>
</tbody>
</table>

INITIALIZE command

The scenarios in this section show how to use the INITIALIZE command in conjunction with key related keywords and subcommands.

Table 82  INITIALIZE scenarios

<table>
<thead>
<tr>
<th>Primary command/scenario task</th>
<th>Subcommand/keyword</th>
<th>Concept/process</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INITIALIZE</td>
<td>IC</td>
<td>initialize areas with basic INITIALIZE command set</td>
<td>419</td>
</tr>
<tr>
<td>Initialize All Areas for a Database</td>
<td>IC with STACK_NAME</td>
<td>create stacked image copy for initialized areas</td>
<td></td>
</tr>
<tr>
<td>INITIALIZE</td>
<td>ALLOCATE</td>
<td>reuse area defined by IAREA on primary command</td>
<td>419</td>
</tr>
<tr>
<td>Initialize and Overwrite an Existing Area</td>
<td>ACTUATE CONFIGURE_AREA SPACE / VOLSER IDCAMS_OPTION</td>
<td>define new VSAM cluster if area does not exist</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>specify size of three separate extents</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>define space allocation unit</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>specify optional IDCAMS definition values</td>
<td></td>
</tr>
</tbody>
</table>
Initialize all areas for a database

The area data sets must be newly defined before the database initialization step. The area data sets and ACB library are dynamically allocated.

Figure 293  JCL to initialize all areas for a database

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-03</td>
<td>Comment to indicate that the IDCAMS define for the new areas must be performed before executing the INITIALIZE command.</td>
</tr>
<tr>
<td>04-07</td>
<td>EXEC and STEPLIB DD statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>08</td>
<td>IMSACB DD statement used for database processing.</td>
</tr>
<tr>
<td>09</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>10</td>
<td>The database PFPSAMP is initialized. All areas of PFPSAMP are initialized by default.</td>
</tr>
<tr>
<td>11-13</td>
<td>An image copy data set is created for each initialized area. These data sets are stacked onto tape.</td>
</tr>
</tbody>
</table>

Initialize and overwrite an existing output area

This technique can also be used on a CHANGE or RELOAD command.

Figure 294  JCL to initialize and overwrite an existing output area (part 1 of 2)
Initialize and overwrite an existing output area

Table 84  Descriptive text for JCL to initialize and overwrite an existing output area

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB DD statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05</td>
<td>The DBD from the IMSACB DD statement is used for database processing.</td>
</tr>
<tr>
<td>06</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>07</td>
<td>Area PFPSAMP1 in database PFPSAMP is initialized.</td>
</tr>
<tr>
<td>08</td>
<td>Area PFPSAMP1 is allocated.</td>
</tr>
<tr>
<td>09</td>
<td>INITIALIZE will reuse the current area defined by IAREA if it already exists.</td>
</tr>
<tr>
<td>10</td>
<td>If this area does not exist, then a new VSAM cluster will be defined. The CONFIGURE_AREA keyword specifies the size of three separate extents for the entire area.</td>
</tr>
<tr>
<td>11</td>
<td>The SPACE keyword specifies tracks as the allocation units.</td>
</tr>
<tr>
<td>12</td>
<td>The VOLSER keyword specifies the volumes to be used for the three extents that are specified on the CONFIGURE_AREA keyword.</td>
</tr>
<tr>
<td>13</td>
<td>The parameter that is specified on the IDCAMS_OPTION keyword specifies additional IDCAMS cluster definition values.</td>
</tr>
</tbody>
</table>

Figure 295 shows how area PFPSAMP1 will be divided across three volumes, based on the parameters that are specified for the CONFIGURE_AREA keyword.
**PROCESS_EPR command**

The scenarios in this section show how to use the PROCESS_EPR command in conjunction with key related keywords and subcommands. Collectively, these examples demonstrate several options that are available for initializing, viewing, and maintaining the Fast Path Analyzer/EP repository.
Create and initialize a repository catalog

Create and initialize a repository catalog

JCL for this scenario is contained in the Fast Path/EP sample library in the member #INITEPR.

Figure 296 JCL to create and initialize a repository catalog (part 1 of 2)

```
01//IDCAMS EXEC PGM=IDCAMS
02//SYSPRINT DD SYSOUT=*  
03//SYSOUT DD SYSOUT=*  
04//SYSIN DD *  
05 DELETE BMC.PFP.PFPEPR  
06 SET MAXCC=0  
07 DEFINE CLUSTER - 
08 (NAME(BMC.PFP.PFPEPR) - 
09 UNIQUE - 
10 SHAREOPTIONS(2) - 
11 CYL(5 1) - 
12 VOL(vvvvvv)) - 
13 DATA(NAME(BMC.PFP.PFPEPR.DATA) - 
14 KEYS(32 0) - 
15 CONTROLINTERVALSIZE(4096) - 
16 FREESPACE(20 10) - 
17 RECORDSIZE(512 512)) - 
18 INDEX(NAME(BMC.PFP.PFPEPR.INDEX) - 
```
Add and delete allocation rules

Figure 296  JCL to create and initialize a repository catalog (part 2 of 2)

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-03</td>
<td>EXEC and output message DD statements for executing IDCAMS program.</td>
</tr>
<tr>
<td>04</td>
<td>SYSIN DD for IDCAMS control statements.</td>
</tr>
<tr>
<td>05-20</td>
<td>Defines and creates the repository VSAM cluster.</td>
</tr>
<tr>
<td>21</td>
<td>End of IDCAMS control card input.</td>
</tr>
<tr>
<td>22-24</td>
<td>EXEC and STEPLIB statements for execution of repository program (PFPEPR00).</td>
</tr>
<tr>
<td>26</td>
<td>Initializes the repository catalog by adding a global allocation rule with defaults DSNAME=NULLFILE and UNIT=SYSDA. When databases are analyzed by other jobs that refer to this repository catalog, only summary analysis statistics will be stored.</td>
</tr>
</tbody>
</table>

Add and delete allocation rules

Figure 297  JCL to add and delete allocation rules (part 1 of 2)

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01///PFPINIT EXEC PGM=PFPEPR00</td>
<td></td>
</tr>
<tr>
<td>02///STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD</td>
<td></td>
</tr>
<tr>
<td>03/// DD DISP=SHR,DSN=IMSVS.RESLIB</td>
<td></td>
</tr>
<tr>
<td>04///PFPSYSIN DD *</td>
<td></td>
</tr>
<tr>
<td>05 PROCESS_EPR REPOSITORY_DSNAM='BMC.PFP.PFPEPR'</td>
<td></td>
</tr>
<tr>
<td>06 MODIFY GLOBAL_ALLOCATION,</td>
<td></td>
</tr>
<tr>
<td>07 DSNAM='PFP.&amp;AREA.&amp;DATE.&amp;TIME',</td>
<td></td>
</tr>
<tr>
<td>08 UNIT=SYSDA</td>
<td></td>
</tr>
<tr>
<td>09 ADD DBD_ALLOCATION,</td>
<td></td>
</tr>
<tr>
<td>10 GROUP_KEY=TEST,</td>
<td></td>
</tr>
<tr>
<td>11 DBD_KEY=PFPSAMP,</td>
<td></td>
</tr>
<tr>
<td>12 DSNAM=NULLFILE,</td>
<td></td>
</tr>
<tr>
<td>13 UNIT=SYSDA</td>
<td></td>
</tr>
<tr>
<td>14 DELETE AREA_ALLOCATION,</td>
<td></td>
</tr>
<tr>
<td>15 SELECT_GROUP=TEST,</td>
<td></td>
</tr>
<tr>
<td>16 SELECT_DBD=PFPSAMP,</td>
<td></td>
</tr>
<tr>
<td>17 SELECT_AREA=PFPSAMP2</td>
<td></td>
</tr>
</tbody>
</table>
List statistics data sets stored in repository

Figure 297  JCL to add and delete allocation rules (part 2 of 2)

18    LIST ALLOCATION
/*

Table 87  Descriptive text for JCL to add and delete allocation rules

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-03</td>
<td>EXEC and STEPLIB statements for execution of repository program (PFPEPR00).</td>
</tr>
<tr>
<td>04</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>05</td>
<td>Specifies the repository catalog data set to be modified</td>
</tr>
<tr>
<td>06-08</td>
<td>Replace the data set name and unit defined for the global allocation rule.</td>
</tr>
<tr>
<td>09-13</td>
<td>This command adds a specific DBD allocation rule for the data base PFPSAMP in the group TEST. The NULLFILE value that is specified for the dsname means that only summary statistics information will be retained for this database.</td>
</tr>
<tr>
<td>14-17</td>
<td>This command deletes any area allocation rule defined for the area PFSPAMP2, in the data base PFPSAMP, for the group TEST.</td>
</tr>
<tr>
<td>18</td>
<td>Requests that all allocation rules in the repository catalog be listed.</td>
</tr>
</tbody>
</table>

List statistics data sets stored in repository

Figure 298  JCL to list statistics data sets stored in repository

01//PFPINIT EXEC PGM=PFPEPR00
02//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
03// DD DISP=SHR,DSN=IMSVS.RESLIB
04//PFPSYSIN DD *
05 PROCESS_EPR REPOSITORY_DSNAME='BMC.PFP.PFPEPR'
06    LIST STATISTICS
/*

Table 88  Descriptive text for JCL to list statistics data sets stored in repository

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-03</td>
<td>EXEC and STEPLIB statements for execution of repository program (PFPEPR00).</td>
</tr>
<tr>
<td>04</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>05</td>
<td>Specifies the repository catalog data set to be modified.</td>
</tr>
<tr>
<td>06</td>
<td>Requests that all statistics records in the repository catalog be listed.</td>
</tr>
</tbody>
</table>
OPTIONS command

The scenarios in this section show how to use the OPTIONS command to request Fast Path Analyzer/EP repository processing for the specified primary command.

Table 89 OPTIONS scenarios

<table>
<thead>
<tr>
<th>Primary command/scenario task</th>
<th>Subcommand/keyword</th>
<th>Concept/process</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPTIONS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Request Repository Processing for REORGANIZE Command</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Store Analysis Statistics in Repository</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPTIONS</td>
<td>REPOSITORY_DSNAME</td>
<td>activate repository processing by using OPTIONS command</td>
<td>425</td>
</tr>
<tr>
<td></td>
<td>REPOSITORY_GROUP</td>
<td>specify name of repository and repository group</td>
<td></td>
</tr>
<tr>
<td></td>
<td>REPOSITORY_RETENTION_COUNT</td>
<td>request overwriting of existing statistics data set</td>
<td></td>
</tr>
<tr>
<td></td>
<td>REPOSITORY_OVERWRITE</td>
<td>specify maximum number of repository statistics catalog entries that will remain recorded within the repository catalog</td>
<td></td>
</tr>
<tr>
<td></td>
<td>REORGANIZE command</td>
<td>store default analysis statistics gathered during reorganization process in repository</td>
<td></td>
</tr>
<tr>
<td>OPTIONS</td>
<td>PFPEPR DD statement</td>
<td>specify repository name using PFPEPR DD statement</td>
<td>427</td>
</tr>
<tr>
<td></td>
<td>REPOSITORY_OVERWRITE</td>
<td>activate repository</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ANALYZE command</td>
<td>request overwriting of existing statistics data set</td>
<td></td>
</tr>
</tbody>
</table>

Request repository processing for REORGANIZE command and store analysis statistics in a repository

The following scenario shows how to request the retention of analysis statistics in the Fast Path Analyzer/EP statistics repository in conjunction with a primary command.

Figure 299 JCL to request repository processing for REORGANIZE command and store analysis statistics in a repository (part 1 of 2)

```plaintext
01 //PFPMAIN EXEC PGM=PFPMAIN,REGION=OM
02 //STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
03 // DD DISP=SHR,DSN=IMSVS.RESLIB
04 // DD DISP=SHR,DSN=IMSVS.DFSMDA
05 // IMSACB DD DISP=SHR,DSN=IMSVS.ACBLIB
06 //PFPSAMP1 DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP1
```
Request repository processing for REORGANIZE command and store analysis statistics in a repository

Figure 299  JCL to request repository processing for REORGANIZE command and store analysis statistics in a repository (part 2 of 2)

```
07//PFPSAMP2 DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP2
08//PFPSAMP3 DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP3
09//PFPOPTS DD *
10 OPTIONS
11   REPOSITORY_DSNAME='BMC.PFP.PFPEPR',
12   REPOSITORY_GROUP='TEST',
13   REPOSITORY_RETENTION_COUNT=5,
14   REPOSITORY_OVERWRITE=YES
15//PFPSYSIN DD *
16   REORGANIZE DBD=PFPSAMP,IAREA=ALL
/*
```

Table 90  Descriptive text for JCL to request repository processing for REORGANIZE command and store analysis statistics in a repository

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05</td>
<td>IMSACB DD statement used for database processing.</td>
</tr>
<tr>
<td>06-08</td>
<td>Specifies the data base areas to be reorganized and analyzed.</td>
</tr>
<tr>
<td>09</td>
<td>PFPOPTS DD for Fast Path/EP OPTIONS statement.</td>
</tr>
<tr>
<td>10</td>
<td>OPTIONS command is specified to request repository processing and associated parameters (lines 11-15). These parameters apply to repository processing for this job execution only.</td>
</tr>
<tr>
<td>11</td>
<td>Defines the repository data set to be used.</td>
</tr>
<tr>
<td>12</td>
<td>Identify the group from which the area’s allocation rules should be derived.</td>
</tr>
<tr>
<td>13</td>
<td>Specifies that up to five statistics data sets for these data base areas will be stored in the repository catalog at any one time.</td>
</tr>
<tr>
<td>14</td>
<td>Indicates that if the statistics data set name defined for these areas exists, it will be overwritten with the new statistics data set created by this job.</td>
</tr>
<tr>
<td>16</td>
<td>All areas of the database are reorganized. If you have a license for the BMC Software Fast Path Analyzer/EP product quick (checksum) pointer validation is performed on all areas by default. This pointer validation creates the statistics for database PFPSAMP to be stored in the repository that is identified by line 11.</td>
</tr>
</tbody>
</table>
The PFPEPR DD statement indicates the repository catalog to be used for the retention of statistics from the analysis process. This job also overwrites any previously existing statistics data set by the same name. An allocation rule has been previously set up in the repository to create statistics data sets using the data set name mask ‘PFP.&IMSID.&DBD.&AREA’. For more information, see Chapter 11, “Statistics repository facility.”

### Figure 300  JCL to request repository processing for ANALYZE command and store analysis statistics in a repository

```plaintext
01//PFP EXEC PGM=PFPMAIN,REGION=OM
02//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
03//          DD DISP=SHR,DSN=IMSVS.RESLIB
04//          DD DISP=SHR,DSN=IMSVS.DFSMDA
05//IMSACB  DD DISP=SHR,DSN=IMSVS.ACBLIB
06//PFPSAMP1 DD DISP=SHR,DSN=BMCPFPP.PFPSAMP.PFPSAMP1
07//PFPSAMP2 DD DISP=SHR,DSN=BMCPFPP.PFPSAMP.PFPSAMP2
08//PFPSAMP3 DD DISP=SHR,DSN=BMCPFPP.PFPSAMP.PFPSAMP3
09//PFPEPR  DD DSN=BMC.PFP.EPR,DISP=SHR
10//PFPOPTS DD   *
11 OPTIONS
12 REPOSITORY_OVERWRITE=YES
13/*
14//PFPSYSIN DD   *
15 ANALYZE DBD=PFPSAMP,IAREA=ALL
/*
```

### Table 91  Descriptive text for JCL to request repository processing for ANALYZE command and store analysis statistics in a repository (part 1 of 2)

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB DD statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05</td>
<td>IMSACB DD statement used for database processing.</td>
</tr>
<tr>
<td>06-08</td>
<td>Area data set names.</td>
</tr>
<tr>
<td>09</td>
<td>PFPEPR DD statement specifies the name of the repository data set to be used.</td>
</tr>
<tr>
<td>10-12</td>
<td>PFPOPTS DD is required for the subsequent OPTIONS command, which is used to define default parameters for the repository. In this case, it defines a temporary override for the repository (REPOSITORY_OVERWRITE=YES).</td>
</tr>
<tr>
<td>12</td>
<td>This job overwrites any previously existing statistics data set which has the same name as that generated by the data set name mask that is specified for the DSNAME allocation rule in the repository catalog.</td>
</tr>
<tr>
<td>13</td>
<td>End of control card input.</td>
</tr>
</tbody>
</table>
Offline EXTEND command

The scenarios in this section show how to use the EXTEND command in conjunction with key related keywords and subcommands.

Before executing the control statement, you must ensure that adequate space is present on the primary volume of the target area data set to accommodate the request for additional UOWs. If the necessary space is not present, allocate volumes to the area data set by using IDCAMS ALTER ADDVOLUME.

Table 92  EXTEND scenarios

<table>
<thead>
<tr>
<th>Primary command/scenario task</th>
<th>Subcommand/keyword</th>
<th>Concept/process</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXTEND</td>
<td>POINTER_VALIDATION IC</td>
<td>■ perform one-step, concurrent maintenance and analysis tasks</td>
<td>429</td>
</tr>
<tr>
<td>Simultaneously Extend IOVF,</td>
<td></td>
<td>■ create image copy of extended DEDB</td>
<td></td>
</tr>
<tr>
<td>Analyze, and Create Image</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Copy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXTEND</td>
<td>POINTER_VALIDATION IC</td>
<td>■ disable automatic pointer validation (must be turned off by user)</td>
<td>429</td>
</tr>
<tr>
<td>Extend IOVF and SDEP</td>
<td>with STACK_NAME</td>
<td>■ create stacked image copies of extended DEDB</td>
<td></td>
</tr>
<tr>
<td>Disable Pointer Validation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stack Image Copies</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXTEND</td>
<td>IAREA</td>
<td>■ limit space extend and associated tasks to one area only</td>
<td>430</td>
</tr>
<tr>
<td>Extend IOVF</td>
<td>EXTEND_IOVF IC</td>
<td>■ extend IOVF</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ create output image copy of extended area</td>
<td></td>
</tr>
</tbody>
</table>

NOTE

Be sure to revise the DBD to increase IOVF so that subsequent offline processes (such as CHANGE or UNLOAD/RELOAD) do not use the old ACB and decrease the amount of IOVF space.
Simultaneously extend IOVF, analyze, and create image copies

Figure 301  JCL to simultaneously extend IOVF, analyze, and create image copies

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB DD statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05</td>
<td>IMSACB DD statement used for database processing.</td>
</tr>
<tr>
<td>06-08</td>
<td>Area data set names.</td>
</tr>
<tr>
<td>09</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>10</td>
<td>Adding 100 units of work to IOVF increases space for the IOVF portion.</td>
</tr>
<tr>
<td>11</td>
<td>A complete analysis (full pointer validation) of all areas is performed on the DEDB.</td>
</tr>
<tr>
<td>12-13</td>
<td>An output image copy is created in the same job step as the EXTEND. The DBD and area name are substituted in the image copy data set name to create a unique name for each area.</td>
</tr>
</tbody>
</table>

Extend IOVF and SDEP, disable automatic analysis, and stack image copies

Figure 302  JCL to extend IOVF and SDEP, disable automatic analysis, and stack image copies (part 1 of 2)
Extend IOVF

Figure 302  JCL to extend IOVF and SDEP, disable automatic analysis, and stack image copies
(part 2 of 2)

```
07//PFPSAMP2  DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP2
08//PFPSAMP3  DD DISP=OLD,DSN=BMCPFP.PFPSAMP.PFPSAMP3
09//PFPSYSIN DD *
10  EXTEND DBD=PFPSAMP,AREA=ALL,EXTEND_IOVF=(UNITS_OF_WORK,100),
     EXTEND_SDEP=(CONTROL_INTERVALS,500),
11     POINTER_VALIDATION=NONE
12   IC STACK_NAME=NAME1,
13   UNIT=TAPE,DISP=(NEW,CATLG),
14   DSNAME='PFP.ICOPY.&DBD.&AREA(+1)' /*
```

Table 94  Descriptive text for JCL to extend IOVF and SDEP, disable automatic analysis, and stack image copies

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB DD statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05</td>
<td>IMSACB DD statement used for database processing.</td>
</tr>
<tr>
<td>06-08</td>
<td>Area data set names.</td>
</tr>
<tr>
<td>09</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>10</td>
<td>All areas of the DEDB are extended. Adding 100 units of work to IOVF and 500 control intervals to SDEP increases the IOVF and SDEP portions.</td>
</tr>
<tr>
<td>11</td>
<td>Pointer validation and physical validation are not performed. POINTER_VALIDATION=NONE overrides the default checksum validation.</td>
</tr>
<tr>
<td>12-14</td>
<td>An image copy data set is created for each area in the database. These data sets are stacked onto tape.</td>
</tr>
<tr>
<td>13</td>
<td>All image copy data sets that are part of the same stack group are written in the order processed onto the same tape volume (or volumes) as file number 1, 2, and 3.</td>
</tr>
<tr>
<td>14</td>
<td>Dynamic allocation is always used for stacked image copy data sets. The DBD and area name are substituted in the image copy data set name to create a unique name for each area.</td>
</tr>
</tbody>
</table>

Extend IOVF

Figure 303  JCL to extend IOVF (part 1 of 2)

```
01//PFP EXEC PGM=PFPMAIN,REGION=DM
02//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
03//   DD DISP=SHR,DSN=IMSVS.RESLIB
04//   DD DISP=SHR,DSN=IMSVS.DFSMDA
05//IMSACB DD DISP=SHR,DSN=IMSVS.ACBLIB
06//PFPSYSIN DD *
07  EXTEND DBD=PFPSAMP,AREA=PFPSAMP1,
08     POINTER_VALIDATION=QUICK,
09  EXTEND_IOVF=(UNITS_OF_WORK,50)
10   IC DSNAME='PFP.ICOPY.&DBD.&AREA(+1)',
```
Figure 303  JCL to extend IOVF (part 2 of 2)

Table 95  Descriptive text for JCL to extend IOVF

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-04</td>
<td>EXEC and STEPLIB DD statements for Fast Path/EP offline execution.</td>
</tr>
<tr>
<td>05</td>
<td>IMSACB DD statement used for database processing.</td>
</tr>
<tr>
<td>06</td>
<td>PFPSYSIN DD for Fast Path/EP control statements.</td>
</tr>
<tr>
<td>07</td>
<td>Area PFPSAMP1 is identified on the EXTEND statement, and is dynamically allocated from its dynamic allocation member in the IMSVS.DFSMDA data set in the STEPLIB concatenation.</td>
</tr>
<tr>
<td>08</td>
<td>If you have a license for the BMC Software Fast Path Analyzer/EP product, quick (checksum) pointer validation is performed on area PFPSAMP1 by default. This pointer validation is performed by default, even if you do not specify it explicitly.</td>
</tr>
<tr>
<td>09</td>
<td>Adding 50 units of work to IOVF increases space for IOVF. Increasing space in this manner allows you to perform a DBD change of the UOW value at a later time.</td>
</tr>
<tr>
<td>10-11</td>
<td>During the area extend, an output image copy is taken simultaneously using standard storage and disposition parameters. The DBD and area name are substituted in the image copy data set name to create a unique name for each area.</td>
</tr>
</tbody>
</table>
Index

A
ACCESS keyword 93
active SDEP boundaries 99
ACTUATE keyword 143, 208, 272, 385, 387, 408, 420
ADD subcommand 324
adding or deleting subset pointers 128
ALLOCATE subcommand 135, 141, 200, 206, 270
AMS_OPTION keyword 144, 209, 273
analysis function
analyze an entire DEDB 95
analyze specific areas 95
analyzing an image copy 92
definition 37
during DEDB change or reload 116
input caching 113
keywords and subcommands 93
performing offline analysis 90
reports 104, 118
sample scenarios 369
selecting reports 104
ANALYZE command 88
analyzing an area while reorganizing 66, 281
analyzing an image copy 92
APF authorization 39, 44
Area Change Modeling Utility 110
Area Change Report 133
area data set allocation 95, 246
Area Initialization Report 266
area integrity, verifying 96
AREA keyword 175
Area Summary Report 119
AREA_KEY keyword 312
areaname DD statement 178

B
BMC Software Support
diagnostic commands 341
DUMP facility 341
Problem Documentation form 341
Problem Summary form 340
system dump example 339
BMC Software, contacting 2
BREAK keyword 262, 415
buffer usage with REORG 65, 280
BYPASS_RECORD keyword 135, 160, 176, 194, 201, 224

C
caching
change function 163
in offline mode 51
IOVF 113
reload function 231
SDEP 113
unload function 197
CHANGE command 74, 286
change function
caching 163
change an entire DEDB 136
change specific areas 137
definition 37
keywords and subcommands 134
preparation 127
restrictions 131
sample scenarios 374
changes to product 32
command set examples
ALLOCATE subcommand 387, 388, 408, 420
ANALYZE 369
CHANGE 374
changing compression 377, 395
concurrent index build 383, 402
EXCLUDE subcommand on EXTRACT 410
EXTEND 428
EXTRACT 409
IC (image copy) subcommand 365, 368, 376, 380, 393, 396, 399, 403, 406, 419, 430
INCLUDE subcommand on EXTRACT 411
INITIALIZE 418
offline DEDB data extract function 409
OFILECTL subcommand on EXTRACT 410, 411
OFILECTL subcommand on UNLOAD 403, 405
OPTIONS 425
OUTPUT subcommand on EXTRACT 410, 412, 414
OUTPUT subcommand on RELOAD 398
PROCESS_EPR 421
RELOAD 390
REORGANIZE 363
command set examples (continued)
  REPORT subcommand 373
  requesting analysis reports 373
  UNLOAD 390
  USER_RECORD subcommand on EXTRACT 415
command sets 38
commands
  ANALYZE 88
  CHANGE 74, 286
  DUMP 341
  EXTEND 278
  EXTRACT 236
  GLOBAL 53
  INITIALIZE 265
  OPTIONS 318, 356
  PFPOPTS OPTIONS 56
  RELOAD 74, 169, 286
  REORGANIZE 62
  RETRIEVE 327
  UNLOAD 74, 169, 286
  COMPRESS keyword 68, 73, 176, 185, 201, 218, 396
 compressed segment data 185, 217
  CONFIGURE_AREA keyword 143, 208, 272
  CONFIGURE_IOVF keyword 144, 209, 273
  CONFIGURE_RAA keyword 144, 209, 272
  CONFIGURE_SDEP keyword 144, 209, 273
console commands 342
conventions, documentation 31
converting DEDBs to MSDBs 188
CORRECTIONS_FILECTL subcommand 93, 100
creating an output image copy while reorganizing 66, 281
customer support 3

database change restrictions 131
Dataclas keyword 312
DBD keyword 68, 93, 134, 136, 175, 176, 200, 202, 268, 283, 295, 312, 322
DBD KEY keyword 312
DBRC
  during change function 133
  during initialization function 267
  during offline analysis function 91
  during reload function 199
  during reorganize function 67, 282
  during unload function 175
  keyword 48
DBRC use 48
DDNAME keyword 93, 102, 249
DEDB restructuring options 127
delete function keywords 312
DELETE subcommand 312, 326
DETAIL_LEVEL keyword 312, 314, 315, 321
DFSRRC00 44
diagnostic DUMP facility 341
diagnostic procedures 335
direct mode 113
DISCARD_FILECTL subcommand 201, 226, 227
DMAC block contents 89
DMB 46
documentation related 30
DOVF_FREESPACE_PERCENT keyword 109, 370
DSNAME keyword 93, 103, 249, 312, 323
DUMP command 341
DUMP console command 342
DUMP facility 341
DUMP_TYPE keyword 341, 342
dynamic allocation 46, 47
electronic documentation 29
error queue elements (EQEs) 68, 91, 175, 282
error tolerance 83, 195
ERROR_THRESHOLD keyword 68, 83, 176, 195
exception conditions, monitoring 109
EXCEPTION_LIMIT keyword 201, 210, 226, 227, 228
EXCLUDE subcommand 135, 155, 176, 189, 219, 251
exclusive access (EX) 199
EXPAND keyword 135, 151, 176, 185, 201, 217, 259, 378, 380, 395, 398, 410, 412, 414
expanding compressed segment data 185, 217
EXPDT keyword 312
expressions 342
EXTEND command 278
extend function
  definition 37
  extend an entire DEDB 284
  extend specific areas 284
  extending IOVF 75, 286
  extending SDEP 77, 288
  keywords and subcommands 283
  restrictions 278
  sample scenarios 428
extend IOVF 74, 169
extend SDEP 74, 169
extend the IOVF and SDEP 278
EXTEND_IOVF keyword 283
EXTEND_IOVF_UOWS keyword 68, 368, 430
EXTEND_SDEP keyword 283
EXTEND_SDEP_CIS keyword 68
EXTRACT command 236
extract file 112
extract file format 242
extract formatting options 237
extract function
  definition 236
  extract data from an entire DEDB 245
  extract data from specific areas 245
extract function (continued)
  keywords and subcommands 239
  sample scenarios 409
EXTRACT_FORMAT keyword 241, 410, 411, 414, 415
extracting data from an offline DEDB 236
extracting segment data 236

F

FABEUR6 Reload File Create Interface 233
FABEUR7 Read Unloaded Database Interface 233
Fast Path Analyzer/EP reports 104
  excluding selected reports 105
  producing only selected reports 106
Fast Path Offline Suite products
  features summary 38
  product capabilities 34
FIELDS keyword 158, 193, 259, 262, 381, 398, 412, 414, 415
FORMAT keyword 176, 188
FRAGMENTATION_PERCENT keyword 68, 72
  free space threshold 109
  FREESPACE_ANALYSIS keyword 104, 373
frequency of segment occurrence 157, 192, 222, 255
full validation reports 106

G

GDG (generation data group) 181, 248, 299, 309
GLOBAL command 53
GROUP_KEY keyword 312

H

header records 262
Help, online 29
historical reports 327
historical summary records 112
history extract file 112
history file 112
HISTORY_DDNAME keyword 53, 94, 112, 323, 328

I

I/O threshold 109
IAREA keyword 68, 69, 93, 94, 134, 136, 176, 200, 202, 245,
  268, 283, 284, 295
IBM DEDB Initialization Utility 265
IBM HD Reorganization Reload Utility 169, 188
IBM HD Reorganization Unload Utility 188, 243
IBM MSDB Maintenance Utility 188
IC subcommand 53, 69, 85, 135, 176, 196, 201, 230, 283, 291
ICACHE keyword 94, 113, 135, 176, 264
IDCAMS 141, 206, 270, 367, 375, 385, 386, 393, 395, 398,
  401, 403, 406, 418, 419, 420, 422, 428
image copy function
  during analysis 295
  during change 162, 296
  during reload 230, 297
  during reorganization 85, 291, 296
  during unload 196, 297
  keywords and subcommands 295
sample scenarios 365, 368, 376, 380, 393, 396, 399, 403,
  406, 419, 430
  stacking image copies 299
implicit values 53
IMS ACB library 46
IMS RESLIB 39
IMSACB DD statement 174
IMSACB library 137
INCLUDE subcommand 135, 154, 176, 189, 219, 251
INDEX keyword 384, 403
INDEX_THREADS keyword 201
 initialization function
  definition 265
  initialize an entire DEDB 269
  initialize specific areas 269
  keywords and subcommands 268
  restrictions 266
  sample scenarios 418
INITIALIZE command 265
input caching
  analysis function 113
  change function 164
  unload function 197
INPUT_DSN_MASK keyword 47, 68, 93, 135, 176, 200, 268,
  283, 373, 381, 398, 401, 403, 414
INPUT_THREADS keyword 135, 164, 176, 198, 201
INSERT_LIMIT_COUNT keyword 365
integration with other BMC products 38
IOVF caching 113
IOVF extend 286
IOVF extension 74, 169, 278
IOVF free space usage 145, 211
IOVF_FREESPACE_PERCENT keyword 370
IOVF_LOAD_HWM keyword 135, 145, 200, 211
IOVF_SAVE_THRESHOLD keyword 68, 366
IOVF_SPACE_ANALYSIS keyword 104, 373
IX subcommand 135

K

keywords
  ACCESS 93
  ACTUATE 143, 208, 272, 385, 387, 408, 420
  AMS_OPTION 144, 209, 273
  AREA 175
  AREA_KEY 312
  BREAK 262, 415
  BYPASS_RECORD 135, 160, 176, 194, 201, 224
  COMPRESSION 68, 73, 176, 185, 201, 218, 396
keywords (continued)

CONFIGURE_AREA 143, 208, 272
CONFIGURE_IOVF 144, 209, 273
CONFIGURE_RAA 144, 209, 272
CONFIGURE_SDEP 144, 209, 273
DATACLAS 312
DBD 68, 93, 134, 136, 175, 176, 200, 202, 268, 283, 295, 312, 322
DBD_KEY 312
DBRC 48
DDNAME 93, 102, 249
DETAIL_LEVEL 312, 314, 321
DOVF_FREESPACE_PERCENT 109, 370
DSNAME 93, 103, 249, 312, 323
DUMP_TYPE 341, 342
ERROR_THRESHOLD 68, 83, 176, 195
EXCEPTION_LIMIT 201, 210, 226, 227, 228
EXPAND 135, 151, 176, 185, 201, 217, 259, 378, 380, 395, 398, 410, 412, 414
EXPDT 312
EXTEND_IOVF 283
EXTEND_IOVF_#UOWS 68, 368, 430
EXTEND_SDEP 283
EXTEND_SDEP_#CIS 68
EXTRACT_FORMAT 241, 410, 411, 414, 415
FIELDS 158, 193, 259, 262, 381, 398, 412, 414, 415
FORMAT 176, 188
FRAGMENTATION_PERCENT 68, 72
FREESPACE_ANALYSIS 373
GROUP_KEY 312
HISTORY_DDNAME 53, 94, 112, 323, 328
IAREA 68, 69, 93, 94, 134, 136, 176, 200, 202, 245, 268, 283, 284, 295
ICACHE 94, 113, 135, 176, 264
INDEX 384, 403
INDEX_THREADS 201
INPUT_DSN_MASK 47, 68, 93, 135, 176, 200, 268, 283, 373
INPUT_DSN_MASK key 381, 398, 401, 403, 414
INPUT_THREADS 135, 164, 176, 198, 201
INSERT_LIMIT_COUNT 365
IOVF_FREESPACE_PERCENT 370
IOVF_LOAD_HWM 135, 145, 200, 211
IOVF_SAVE_THRESHOLD 68
IOVF_SAVE_THRESHOLD 366
IOVF_SPACE_ANALYSIS 373
LARGEST_DATABASE_RECORDS 135, 201
LOCATION 365, 382
MESSAGE_NUMBER 56
MESSAGE_LEVEL 56, 350
MESSAGE_LIMIT 56, 59, 350
MESSAGE_NUMBER 350
MESSAGE_OVERRIDE 353
MESSAGE_SUPPRESSION 58, 67, 94, 111, 130, 172, 281, 283
MGMTCLAS 312
MODEL_DD_NAME 373
MODEL_DDNAME 93, 110
OAREA 134, 137, 175, 178, 200, 203
OCACHE 135, 164, 201, 231
ORPHANED_SDEP_MSG 93, 94, 283
OUTPUT_DSN_MASK 47, 135, 176, 200
OUTPUT_DSN_MASK key 381, 398
OUTPUT_THREADS 135, 165
POINTER_DEFAULT 69, 93, 201, 283
RAA_FREESPACE_PERCENT 371
RAP_OVERFLOW_PERCENT 371
RAP_VALIDATION 69, 93, 201, 283
RECORD_IO_AVERAGE 109
RECORD_IO_MAXIMUM 109
RECORD_IOVF_PERCENT 109
REPORT_DDNAME 107
REPORT_DEFAULT 105
REPORT_HEADING 107
REPORT_LINE_COUNT 107
REPOSITORY_DSNAME 318, 350, 356, 368
REPOSITORY_GROUP 319, 368
REPOSITORY_OVERWRITE 320, 369, 372
REPOSITORY_RETENTION_COUNT 325, 369
REPOSITORY_RETENTION_PERIOD 323, 325
RETPD 312
SAMPLE_INTERVAL 157, 176, 192, 201, 222, 255
SAMPLE_LIMIT 157, 176, 192, 201, 222, 255
SDEP_PROCESS 135, 152, 176, 186, 201, 218, 376, 378, 380, 382, 384, 393, 395, 398, 401, 403, 405
SDEP_VALIDATION 69, 93, 99, 116, 162, 201, 230, 283
SEGMENT 176, 201, 365, 381, 382, 398, 401, 410, 411, 412, 414
SEGMENT_RECORD_PREFIX 261, 415
SEGMENT_RECORD_SUFFIX 261
SELECT_AREA 312, 323, 327
SELECT_DATE 323
SELECT_DATE_RANGE 327
SELECT_DBD 312, 327
SELECT_GROUP 312, 323, 327
SELECT_UOW 68, 72, 365, 366
SORT 201, 263, 393, 412
SORT_OPTION 201, 263
STACK_NAME 299, 365, 376, 406, 419, 430
STARTUOW 106
STOPUOW 106
STORCLAS 312
SUBSET_POINTERS 135, 154, 176, 187
SYNONYM_CHAIN_LENGTH 109
SYNONYM_CHAIN_MAXIMUM 109
SYNONYM_CHAIN_PERCENT 109
TYPE_RUN 283
UNIT 312
UOW_DETAILLED_ANALYSIS 105
UOW_DOVF_PERCENT 109
UOW_IOVF_AVERAGE 109
UOW_IOVF_MAXIMUM 109
UOW_IOVF_PERCENT 109
A   B   C   D   E   F   G   H   I   J   K   L   M   N   O   P   Q   R   S   T   U   V   W   X   Y   Z

keywords (continued)
VOLSER 312
WHERE 156, 159, 176, 190, 193, 201, 221, 224, 252, 256, 260, 365, 382, 410

L
LARGEST_DATABASE_RECORDS keyword 135, 201
LB 99
LE 99
LIST subcommand 350, 353
load control 78, 146, 211
LOADCTL subcommand 68, 78, 135, 146, 201, 212
LOCATION keyword 365, 382
logical SDEPs 152, 186, 218

M
message modification 347
MESSAGE_LEVEL keyword 56, 350
MESSAGE_LIMIT keyword 56, 59, 350
MESSAGE_NUMBER keyword 56, 350
MESSAGE_OVERRIDE keyword 353
MESSAGE_SUPPRESSION keyword 58, 67, 94, 111, 130, 172, 281, 283
messages
changing severity level 56, 353
listing stored message customizations 353
permanent customization options 56, 348
permanently modifying messages 348
restoring overrides 57, 354
suppressing 58, 67, 111, 130, 172, 281, 353
temporarily modifying messages 56
MGMTCLAS keyword 312
MODEL_DD_NAME keyword 373
MODEL_DDNAME keyword 93, 110
Modeling Utility 110
modify function keywords 312
MODIFY subcommand 312
monitoring area characteristics 104
MSDBs (Main Storage Databases) 169
multiple area data sets (MADS) 49, 50
change function 134
initialization function 267
reload function 200
reorganize function 64, 68, 282
unload function 175
multitasking 50

N
NEWACB DD statement 174
NEWACB library 178
NULLFILE 307, 309, 310

O
OAREA keyword 134, 137, 175, 178, 200, 203
OCACHE keyword 135, 164, 201, 231
offline analysis 90
offline DEDB data extract function
controlling content of segment header and trailer records 262
controlling segment expansion 259
controlling the content of segment record prefixes and suffixes 261
controlling the output of segment records 259
DBRC considerations 238
definition 236
including and excluding segments for input 154, 251
input caching 264
inputs and outputs 237
keywords and subcommands 239
limiting generation of output segment records 255
output format options 237
sample scenarios 409
selecting the extract file format 241
sorting the extract file 263
offline mode processing 42
OFILECTL subcommand 176, 247
online Help 29
online IOVF extend 286
online mode processing 42
online SDEP extend 286
OPTIONS command 318, 356, 425
orphaned sequential dependents 100
ORPHANED_SDEP_MSG keyword 93, 94, 283
out-of-space condition 74, 278, 286
output caching 164
output image copy during analysis 114, 274
OUTPUT subcommand 158, 176, 192, 251, 255, 256, 259
OUTPUT_DSN_MASK keyword 47, 135, 176, 200, 381, 398
OUTPUT_THREADS keyword 135, 165
OVERRIDE subcommand 56, 350

P
parallel processing 51, 163, 197
PFPEPR00 utility program 310, 312, 322, 349
PFPPINIT member 266
PFPOPTS DD statement 325
PFPOPTS OPTIONS command 56
PFPSYSIN control statements 44
PFPTOTAL DD statement 140, 180, 205
physical integrity 96
physical SDEPs 186
physical validation 96
placing segments 211
Pointer Analysis Report 120
pointer integrity validation 97
pointer validation 97
POINTER_ANALYSIS keyword 104
POINTER_VALIDATION keyword 69, 93, 201, 283
problems 338
product changes 32
product installation 38
product integration 38
product support 3, 335
program extensions for unload/reload 233
publications
   related 30
publications, related 30

R
RAA_FREESPACE_PERCENT keyword 371
RAP utilization threshold 109
RAP_OVERFLOW_PERCENT keyword 371
RAP_VALIDATION keyword 69, 93, 201, 283
Read Unloaded Database Interface (FABEUR7) 233
reclaim free space 169
RECON data sets 49
Record Length Analysis Report 121
Record Placement Analysis report 122
Record Profile Analysis Report 122
RECORD_IO_AVERAGE keyword 109
RECORD_IO_MAXIMUM keyword 109
RECORD_IOVF_PERCENT keyword 109
RECORD_LENGTH_ANALYSIS keyword 104
RECORD_PLACEMENT_ANALYSIS keyword 104
RECORD_PROFILE_ANALYSIS keyword 104
Recovery Needed condition 199
recovery status 49
REGION size 44
registered area data set 47
related publications 30
RELOAD command 74, 169, 286
Reload File Create Interface (FABEUR6) 233
reload function
   definition 37
   including and excluding segments for input 219
   keywords and subcommands 200
   preparation 170
   reload an entire DEDB 202
   reload specific areas 202
   sample scenarios 390
reorganization function
   definition 37
   extend IOVF and SDEP online 286
   keywords and subcommands 68
   reorganize an entire DEDB 70
   reorganize specific areas 70
   restrictions 63
   sample scenarios 363
REORGANIZE command 62
report examples 118
report formatting 107
report keywords
FREESPACE_ANALYSIS 104
IOVF_SPACE_ANALYSIS 104
POINTER_ANALYSIS 104
RECORD_LENGTH_ANALYSIS 104
RECORD_PLACEMENT_ANALYSIS 104
RECORD_PROFILE_ANALYSIS 104
SEGMENT_IO_ANALYSIS 104
SEGMENT_LENGTH_ANALYSIS 104
SEGMENT_PLACEMENT_ANALYSIS 104
SYNONYM_CHAIN_ANALYSIS 104
UOW_DETAILED_ANALYSIS 104
report routing 107
REPORT subcommand 53, 54, 93, 104, 135, 201, 327
REPORT_DDNAME keyword 107
REPORT_DEFAULT keyword 105
REPORT-heading keyword 107
REPORT_LINE_COUNT keyword 107
reporting area characteristics 104
reports
   Area Change Report 133
   Area Initialization Report 266
   Area Summary Report 119
   full validation 106
   historical repository 327
   Pointer Analysis Report 120
   Record Length Analysis Report 121
   Record Placement Analysis 122
   Record Profile Analysis Report 122
   Segment I/O Analysis Report 120
   Segment Placement Analysis Report 121
   statistical 327
   Synonym Chain Analysis Report 123
   UOW Detailed Analysis Report 120
REPORTS data set 266
repository facility
   activating message customizations 355
   activating repository processing 317
   adding statistics records 324
   allocating statistics data set 319, 320
   controlling retention of entries 325
   creating a repository catalog 311
   creating an additional repository catalog 311
   deleting statistics records 324
   function of allocation rules 308
   inputs and outputs 307
   listing statistics records 323
   managing allocation rules 312
   producing a history file 328
   producing reports 327
   repository catalog 317
   sample scenarios 421
   REPOSITORY_DSNAME keyword 318, 350, 356, 368
   REPOSITORY_GROUP keyword 319, 368
   REPOSITORY_OVERWRITE keyword 320, 369, 372
   REPOSITORY_RETENTION_COUNT keyword 325, 369
   REPOSITORY_RETENTION_PERIOD keyword 323, 325
subcommands (continued)
  LOADCTL 68, 78, 135, 146, 201, 212
  MODIFY 312
  OFILECTL 176
  OUTPUT 158, 176, 192, 251, 255, 256, 259
  OVERRIDE 56, 350
  REPORT 53, 54, 93, 104, 135, 201, 327
  RESET 57, 350
  RETRIEVE 327
  THRESHOLD 53, 93, 109, 135, 370
  USER_RECORD 262, 415
subset pointers
  during change function 154
  during unload function 187
SUBSET_POINTERS keyword 135, 154, 176, 187
summary of changes 32
support, customer 3
Synonym Chain Analysis Report 123
SYNONYM_CHAIN_ANALYSIS keyword 104
SYNONYM_CHAIN_LENGTH keyword 109
SYNONYM_CHAIN_MAXIMUM keyword 109
SYNONYM_CHAIN_PERCENT keyword 109
syntax statement conventions 31
system requirements 39

T
  technical support 3
  thread processing 163, 197, 231
threshold keywords
  DOVF_FREESPACE_PERCENT 109
  RECORD_IO_AVERAGE 109
  RECORD_IO_MAXIMUM 109
  RECORD_IOVF_PERCENT 109
  SYNONYM_CHAIN_LENGTH 109
  SYNONYM_CHAIN_MAXIMUM 109
  SYNONYM_CHAIN_PERCENT 109
  UOW_DOVF_PERCENT 109
  UOW_IOVF_AVERAGE 109
  UOW_IOVF_MAXIMUM 109
  UOW_IOVF_PERCENT 109
threshold settings 109
THRESHOLD subcommand 53, 93, 109, 135, 370
threshold types, analysis 109
thresholds
  free space 109
  I/O 109
  RAP utilization 109
trailer records 262
TYPE_RUN keyword 283

U
  UNIT keyword 312
  UNLOAD command 74, 169, 286
Notes