This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the "Documentation") is for your informational purposes only and is subject to change or withdrawal by CA at any time. This Documentation is proprietary information of CA and may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA.

If you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with “Restricted Rights.” Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2017 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.
Table of Contents

Utility Operations ................................................................. 19
Using the Utility Statements .................................................. 19
  Utility Descriptions ......................................................... 19
  SQL and Non-SQL ........................................................... 22
Central Version Considerations .............................................. 24

Command Facility Considerations .......................................... 26

Utility Statements ................................................................. 28
ARCHIVE JOURNAL .............................................................. 29
  Authorization ............................................................... 29
  Syntax ....................................................................... 29
  Parameters ............................................................... 30
  Usage ....................................................................... 31
  JCL Considerations .................................................... 33
  Example ................................................................. 33
  Sample Output ........................................................ 33
ARCHIVE LOG ................................................................. 34
  Authorization .............................................................. 35
  Syntax ................................................................. 35
  Parameters .............................................................. 35
  Usage ................................................................. 35
  JCL Considerations .................................................... 36
  Example ................................................................. 36
  Sample Output ........................................................ 36
ARCHIVE TRACE ............................................................. 37
  Authorization ............................................................ 37
  Syntax ................................................................. 37
  Parameters .............................................................. 38
  Example ................................................................. 38
BACKUP ............................................................................. 38
  Authorization ............................................................ 39
  Syntax ................................................................. 39
  Parameters .............................................................. 39
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage</td>
<td>40</td>
</tr>
<tr>
<td>JCL Considerations</td>
<td>41</td>
</tr>
<tr>
<td>Examples</td>
<td>41</td>
</tr>
<tr>
<td>Sample Output</td>
<td>42</td>
</tr>
<tr>
<td><strong>BUILD</strong></td>
<td>43</td>
</tr>
<tr>
<td>Authorization</td>
<td>44</td>
</tr>
<tr>
<td>Syntax</td>
<td>44</td>
</tr>
<tr>
<td>Parameters</td>
<td>45</td>
</tr>
<tr>
<td>Usage</td>
<td>47</td>
</tr>
<tr>
<td>JCL Considerations</td>
<td>47</td>
</tr>
<tr>
<td>Examples</td>
<td>48</td>
</tr>
<tr>
<td>Sample Output</td>
<td>49</td>
</tr>
<tr>
<td><strong>CLEANUP</strong></td>
<td>49</td>
</tr>
<tr>
<td>Authorization</td>
<td>50</td>
</tr>
<tr>
<td>Syntax</td>
<td>50</td>
</tr>
<tr>
<td>Parameters</td>
<td>50</td>
</tr>
<tr>
<td>Usage</td>
<td>51</td>
</tr>
<tr>
<td>JCL Considerations</td>
<td>51</td>
</tr>
<tr>
<td>Example</td>
<td>52</td>
</tr>
<tr>
<td>Sample Output</td>
<td>52</td>
</tr>
<tr>
<td><strong>CONVERT CATALOG</strong></td>
<td>52</td>
</tr>
<tr>
<td>Authorization</td>
<td>52</td>
</tr>
<tr>
<td>Syntax</td>
<td>53</td>
</tr>
<tr>
<td>Usage</td>
<td>53</td>
</tr>
<tr>
<td><strong>CONVERT PAGE</strong></td>
<td>53</td>
</tr>
<tr>
<td>Authorization</td>
<td>54</td>
</tr>
<tr>
<td>Syntax</td>
<td>54</td>
</tr>
<tr>
<td>Parameters</td>
<td>54</td>
</tr>
<tr>
<td>Usage</td>
<td>55</td>
</tr>
<tr>
<td>JCL Considerations</td>
<td>57</td>
</tr>
<tr>
<td>Examples</td>
<td>57</td>
</tr>
<tr>
<td>Example 1</td>
<td>57</td>
</tr>
<tr>
<td>Example 2</td>
<td>58</td>
</tr>
<tr>
<td>Example 3</td>
<td>58</td>
</tr>
<tr>
<td>Example 4</td>
<td>58</td>
</tr>
<tr>
<td>Bad Example 4</td>
<td>59</td>
</tr>
<tr>
<td>Sample Output</td>
<td>60</td>
</tr>
<tr>
<td><strong>CREATE DSMODEL</strong></td>
<td>61</td>
</tr>
<tr>
<td>Authorization</td>
<td>61</td>
</tr>
<tr>
<td>Syntax</td>
<td>61</td>
</tr>
<tr>
<td>Parameters</td>
<td>62</td>
</tr>
<tr>
<td>Usage</td>
<td>63</td>
</tr>
<tr>
<td>Command</td>
<td>Example</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
</tr>
<tr>
<td>EXPAND PAGE</td>
<td></td>
</tr>
<tr>
<td>Authorization</td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td></td>
</tr>
<tr>
<td>JCL Considerations</td>
<td></td>
</tr>
<tr>
<td>Example</td>
<td></td>
</tr>
<tr>
<td>Sample Output</td>
<td></td>
</tr>
<tr>
<td>FASTLOAD</td>
<td></td>
</tr>
<tr>
<td>Authorization</td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td></td>
</tr>
<tr>
<td>JCL Considerations</td>
<td></td>
</tr>
<tr>
<td>Example</td>
<td></td>
</tr>
<tr>
<td>Sample Output</td>
<td></td>
</tr>
<tr>
<td>FIX ARCHIVE</td>
<td></td>
</tr>
<tr>
<td>Authorization</td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td></td>
</tr>
<tr>
<td>JCL Considerations</td>
<td></td>
</tr>
<tr>
<td>Example</td>
<td></td>
</tr>
<tr>
<td>Sample Output</td>
<td></td>
</tr>
<tr>
<td>FIX PAGE</td>
<td></td>
</tr>
<tr>
<td>Authorization</td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td></td>
</tr>
<tr>
<td>JCL Considerations</td>
<td></td>
</tr>
<tr>
<td>Examples</td>
<td></td>
</tr>
<tr>
<td>Sample Output</td>
<td></td>
</tr>
<tr>
<td>FORMAT</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>Page</td>
</tr>
<tr>
<td>------------------</td>
<td>------</td>
</tr>
<tr>
<td>Authorization</td>
<td>95</td>
</tr>
<tr>
<td>Syntax</td>
<td>95</td>
</tr>
<tr>
<td>Parameters</td>
<td>96</td>
</tr>
<tr>
<td>Usage</td>
<td>98</td>
</tr>
<tr>
<td>JCL Considerations</td>
<td>103</td>
</tr>
<tr>
<td>Examples</td>
<td>104</td>
</tr>
<tr>
<td>Sample Output</td>
<td>105</td>
</tr>
<tr>
<td>INSTALL STAMPS</td>
<td>106</td>
</tr>
<tr>
<td>Authorization</td>
<td>106</td>
</tr>
<tr>
<td>Syntax</td>
<td>107</td>
</tr>
<tr>
<td>Parameters</td>
<td>107</td>
</tr>
<tr>
<td>Usage</td>
<td>107</td>
</tr>
<tr>
<td>JCL Considerations</td>
<td>108</td>
</tr>
<tr>
<td>Examples</td>
<td>108</td>
</tr>
<tr>
<td>Sample Output</td>
<td>108</td>
</tr>
<tr>
<td>LOAD</td>
<td>109</td>
</tr>
<tr>
<td>Authorization</td>
<td>110</td>
</tr>
<tr>
<td>Syntax</td>
<td>110</td>
</tr>
<tr>
<td>Parameters</td>
<td>111</td>
</tr>
<tr>
<td>Usage</td>
<td>116</td>
</tr>
<tr>
<td>JCL Considerations</td>
<td>117</td>
</tr>
<tr>
<td>Examples</td>
<td>118</td>
</tr>
<tr>
<td>Sample Output</td>
<td>119</td>
</tr>
<tr>
<td>LOCK</td>
<td>123</td>
</tr>
<tr>
<td>Authorization</td>
<td>123</td>
</tr>
<tr>
<td>Syntax</td>
<td>123</td>
</tr>
<tr>
<td>Parameters</td>
<td>123</td>
</tr>
<tr>
<td>Usage</td>
<td>124</td>
</tr>
<tr>
<td>JCL Considerations</td>
<td>124</td>
</tr>
<tr>
<td>Examples</td>
<td>124</td>
</tr>
<tr>
<td>Sample Output</td>
<td>125</td>
</tr>
<tr>
<td>MAINTAIN INDEX</td>
<td>125</td>
</tr>
<tr>
<td>Authorization</td>
<td>126</td>
</tr>
<tr>
<td>Syntax</td>
<td>126</td>
</tr>
<tr>
<td>Parameters</td>
<td>126</td>
</tr>
<tr>
<td>Usage</td>
<td>129</td>
</tr>
<tr>
<td>JCL Considerations</td>
<td>138</td>
</tr>
<tr>
<td>Examples</td>
<td>140</td>
</tr>
<tr>
<td>Sample Output</td>
<td>140</td>
</tr>
<tr>
<td>MERGE ARCHIVE</td>
<td>141</td>
</tr>
<tr>
<td>Authorization</td>
<td>142</td>
</tr>
<tr>
<td>Syntax</td>
<td>142</td>
</tr>
</tbody>
</table>
UNLOCK .................................................................................................................. 261
Parameters ............................................................................................................. 261
Usage ....................................................................................................................... 264
JCL Considerations ............................................................................................... 268
Examples ............................................................................................................... 268
Sample Output ...................................................................................................... 268

SYNCHRONIZE STAMPS ...................................................................................... 269
Syntax ..................................................................................................................... 269
Parameters ............................................................................................................. 270
Usage ....................................................................................................................... 270
JCL Considerations ............................................................................................... 271
Examples ............................................................................................................... 271
Sample Output ...................................................................................................... 271

TUNE INDEX .......................................................................................................... 272
Authorization ........................................................................................................... 273
Syntax ..................................................................................................................... 273
Parameters ............................................................................................................. 274
Usage ....................................................................................................................... 276
JCL Considerations ............................................................................................... 277
Examples ............................................................................................................... 277
Sample Output ...................................................................................................... 278

UNLOAD ................................................................................................................... 278
Authorization .......................................................................................................... 279
Syntax ..................................................................................................................... 279
Parameters ............................................................................................................. 280
Usage ....................................................................................................................... 281
How UNLOAD Works ............................................................................................. 283
The Unload/Reload Subschemas ............................................................................. 284
The Unload/Reload SEGMENT ............................................................................... 285
The Unload/Reload DMCL ...................................................................................... 285
General Procedure for UNLOAD and RELOAD .................................................... 286
JCL Considerations ............................................................................................... 287
Example .................................................................................................................. 287
Sample Output ...................................................................................................... 287

UNLOCK ................................................................................................................... 288
Authorization .......................................................................................................... 289
Syntax ..................................................................................................................... 289
Parameters ............................................................................................................. 289
Usage ....................................................................................................................... 289
JCL Considerations ............................................................................................... 289
Examples ............................................................................................................... 290
Sample Output ...................................................................................................... 290
Utilities Programs (CA IDMS/DB Utilities) ........................................ 300

IDMSCALC Utility ........................................................................... 300
Usage ................................................................................................. 300
Calling the IDMSCALC Routine ....................................................... 300

IDMSDBAN ....................................................................................... 301
Authorization ................................................................................. 302
Syntax ............................................................................................... 302
Input Parameter Statements .............................................................. 303
Usage ................................................................................................. 307
JCL Considerations ....................................................................... 311
Example ............................................................................................ 312
Sample Output ................................................................................ 312

IDMSDIRL ....................................................................................... 316
Syntax ............................................................................................... 317
Input Parameter Statements .............................................................. 317
Usage ................................................................................................. 318
JCL Considerations ....................................................................... 319
Examples ........................................................................................ 320
Sample Output ................................................................................ 320

IDMSLOOK ..................................................................................... 320
Syntax ............................................................................................... 321
Parameters ....................................................................................... 322
Usage ................................................................................................. 325
JCL Considerations ....................................................................... 325

Utilities 12
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Examples</td>
<td>325</td>
</tr>
<tr>
<td>Sample Output</td>
<td>326</td>
</tr>
<tr>
<td>IDMSRPTS</td>
<td>330</td>
</tr>
<tr>
<td>Report Types</td>
<td>330</td>
</tr>
<tr>
<td>Non-database reports</td>
<td>330</td>
</tr>
<tr>
<td>Schema Reports</td>
<td>331</td>
</tr>
<tr>
<td>Subschema reports</td>
<td>331</td>
</tr>
<tr>
<td>Physical Database Definition reports</td>
<td>332</td>
</tr>
<tr>
<td>Syntax</td>
<td>333</td>
</tr>
<tr>
<td>Statements</td>
<td>334</td>
</tr>
<tr>
<td>Usage</td>
<td>342</td>
</tr>
<tr>
<td>JCL Considerations</td>
<td>343</td>
</tr>
<tr>
<td>Examples</td>
<td>343</td>
</tr>
<tr>
<td>Sample Output</td>
<td>344</td>
</tr>
<tr>
<td>IDMSRSTC</td>
<td>349</td>
</tr>
<tr>
<td>Syntax</td>
<td>349</td>
</tr>
<tr>
<td>Input Parameter Statements</td>
<td>350</td>
</tr>
<tr>
<td>Usage</td>
<td>352</td>
</tr>
<tr>
<td>Example</td>
<td>353</td>
</tr>
<tr>
<td>Sample Output</td>
<td>354</td>
</tr>
<tr>
<td>z/OS JCL - IDMSBCF and Utilities</td>
<td>355</td>
</tr>
<tr>
<td>Batch Command Facility (z/OS)</td>
<td>355</td>
</tr>
<tr>
<td>Utility Statements (z/OS)</td>
<td>357</td>
</tr>
<tr>
<td>ARCHIVE JOURNAL</td>
<td>358</td>
</tr>
<tr>
<td>ARCHIVE LOG</td>
<td>358</td>
</tr>
<tr>
<td>ARCHIVE TRACE</td>
<td>359</td>
</tr>
<tr>
<td>BACKUP</td>
<td>359</td>
</tr>
<tr>
<td>BUILD</td>
<td>359</td>
</tr>
<tr>
<td>CLEANUP</td>
<td>360</td>
</tr>
<tr>
<td>CONVERT CATALOG</td>
<td>361</td>
</tr>
<tr>
<td>CONVERT PAGE</td>
<td>361</td>
</tr>
<tr>
<td>EXPAND PAGE</td>
<td>362</td>
</tr>
<tr>
<td>EXTRACT JOURNAL</td>
<td>362</td>
</tr>
<tr>
<td>FASTLOAD</td>
<td>363</td>
</tr>
<tr>
<td>FIX ARCHIVE</td>
<td>364</td>
</tr>
<tr>
<td>FIX PAGE</td>
<td>365</td>
</tr>
<tr>
<td>FORMAT</td>
<td>365</td>
</tr>
<tr>
<td>INSTALL STAMPS</td>
<td>366</td>
</tr>
<tr>
<td>LOAD</td>
<td>367</td>
</tr>
<tr>
<td>LOCK</td>
<td>368</td>
</tr>
</tbody>
</table>
Utility Statements (z/VSE)

- MERGE ARCHIVE ......................................................................................................................... 370
- PRINT INDEX ................................................................................................................................. 370
- PRINT JOURNAL ............................................................................................................................ 370
- PRINT LOG ......................................................................................................................................... 371
- PRINT PAGE ....................................................................................................................................... 371
- PRINT SPACE ..................................................................................................................................... 372
- PRINT TRACE ..................................................................................................................................... 372
- PUNCH ................................................................................................................................................ 372
- RELOAD .............................................................................................................................................. 373
- REORG .............................................................................................................................................. 375
- RESTORE .......................................................................................................................................... 377
- RESTRUCTURE .................................................................................................................................. 377
- RESTRUCTURE CONNECT .................................................................................................................. 378
- ROLLBACK ......................................................................................................................................... 378
- ROLLFORWARD ................................................................................................................................. 378
- SYNCHRONIZE STAMPS ................................................................................................................... 379
- TUNE INDEX ...................................................................................................................................... 380
- UNLOCK ............................................................................................................................................. 380
- UNLOAD ............................................................................................................................................. 380
- UPDATE STATISTICS ......................................................................................................................... 381
- VALIDATE .......................................................................................................................................... 382

Utility Programs (z/OS) ......................................................................................................................... 383
- IDMSDBAN ........................................................................................................................................ 383
- IDMSDIRL ......................................................................................................................................... 384
- IDMSLOOK ......................................................................................................................................... 385
- IDMSRPTS .......................................................................................................................................... 386
- IDMSRSTC .......................................................................................................................................... 387
- IDMSRSTT .......................................................................................................................................... 388

z/VSE JCL ................................................................................................................................................. 389

- COPY Facility ...................................................................................................................................... 389
  - Syntax ............................................................................................................................................. 389
  - Parameters .................................................................................................................................... 389

About the IDMSLRLS Procedure ........................................................................................................... 390

SYSIDMS Parameter File for Execution .............................................................................................. 390

Batch Command Facility (z/VSE) .......................................................................................................... 391

Utility Statements (z/VSE) ...................................................................................................................... 392

- ARCHIVE JOURNAL ......................................................................................................................... 392
- ARCHIVE LOG ................................................................................................................................. 393
- ARCHIVE TRACE ............................................................................................................................. 393
<table>
<thead>
<tr>
<th>Utility Programs (z/VSE)</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACKUP</td>
<td>393</td>
</tr>
<tr>
<td>BUILD</td>
<td>393</td>
</tr>
<tr>
<td>CLEANUP</td>
<td>394</td>
</tr>
<tr>
<td>CONVERT CATALOG</td>
<td>395</td>
</tr>
<tr>
<td>CONVERT PAGE</td>
<td>395</td>
</tr>
<tr>
<td>EXPAND PAGE</td>
<td>396</td>
</tr>
<tr>
<td>EXTRACT JOURNAL</td>
<td>396</td>
</tr>
<tr>
<td>FASTLOAD</td>
<td>397</td>
</tr>
<tr>
<td>FIX ARCHIVE</td>
<td>398</td>
</tr>
<tr>
<td>FIX PAGE</td>
<td>399</td>
</tr>
<tr>
<td>FORMAT</td>
<td>401</td>
</tr>
<tr>
<td>INSTALL STAMPS</td>
<td>402</td>
</tr>
<tr>
<td>LOAD</td>
<td>403</td>
</tr>
<tr>
<td>LOCK</td>
<td>404</td>
</tr>
<tr>
<td>MAINTAIN INDEX</td>
<td>405</td>
</tr>
<tr>
<td>PRINT INDEX</td>
<td>405</td>
</tr>
<tr>
<td>PRINT JOURNAL</td>
<td>406</td>
</tr>
<tr>
<td>PRINT LOG</td>
<td>406</td>
</tr>
<tr>
<td>PRINT PAGE</td>
<td>406</td>
</tr>
<tr>
<td>PRINT SPACE</td>
<td>406</td>
</tr>
<tr>
<td>PRINT TRACE</td>
<td>407</td>
</tr>
<tr>
<td>PUNCH</td>
<td>407</td>
</tr>
<tr>
<td>RELOAD</td>
<td>408</td>
</tr>
<tr>
<td>REORG</td>
<td>409</td>
</tr>
<tr>
<td>RESTORE</td>
<td>411</td>
</tr>
<tr>
<td>RESTRUCTURE</td>
<td>411</td>
</tr>
<tr>
<td>RESTRUCTURE CONNECT</td>
<td>412</td>
</tr>
<tr>
<td>ROLLBACK</td>
<td>412</td>
</tr>
<tr>
<td>ROLLFORWARD</td>
<td>412</td>
</tr>
<tr>
<td>SYNCHRONIZE STAMPS</td>
<td>413</td>
</tr>
<tr>
<td>TUNE INDEX</td>
<td>414</td>
</tr>
<tr>
<td>UNLOCK</td>
<td>414</td>
</tr>
<tr>
<td>UNLOAD</td>
<td>414</td>
</tr>
<tr>
<td>UPDATE STATISTICS</td>
<td>415</td>
</tr>
<tr>
<td>VALIDATE</td>
<td>416</td>
</tr>
<tr>
<td>IDMSDBAN</td>
<td>417</td>
</tr>
<tr>
<td>IDMSDIRL</td>
<td>418</td>
</tr>
<tr>
<td>IDMSLOOK</td>
<td>419</td>
</tr>
<tr>
<td>IDMSRPTS</td>
<td>420</td>
</tr>
<tr>
<td>IDMSRSTC</td>
<td>421</td>
</tr>
<tr>
<td>IDMSRSTT</td>
<td>423</td>
</tr>
</tbody>
</table>
CMS Commands - IDMSBCF and Utilities .................................................. 428

Batch Command Facility (CMS Commands) ............................................. 428

Utility Statements (CMS Commands) ...................................................... 429

ARCHIVE JOURNAL ........................................................................... 430
ARCHIVE LOG ................................................................................... 431
BACKUP ............................................................................................ 431
BUILD ............................................................................................... 432
CLEANUP .......................................................................................... 433
CONVERT CATALOG .......................................................................... 433
CONVERT PAGE ................................................................................ 434
EXPAND PAGE .................................................................................. 434
EXTRACT JOURNAL .......................................................................... 435
FASTLOAD .......................................................................................... 435
FIX ARCHIVE ..................................................................................... 436
FIX PAGE ............................................................................................ 436
FORMAT ........................................................................................~~~~ 437
INSTALL STAMPS ............................................................................. 438
LOAD .................................................................................................... 438
LOCK ................................................................................................... 439
MAINTAIN INDEX ............................................................................... 440
MERGE ARCHIVE ............................................................................ 441
PRINT INDEX ..................................................................................... 441
PRINT JOURNAL ................................................................................ 442
PRINT LOG ........................................................................................ 442
PRINT PAGE ....................................................................................... 442
PRINT SPACE .................................................................................... 443
PUNCH ................................................................................................ 444
RELOAD ............................................................................................... 445
RESTORE ............................................................................................. 446
RESTRUCTURE ................................................................................... 447
RESTRUCTURE CONNECT ................................................................. 447
ROLLBACK ......................................................................................... 448
ROLLFORWARD .................................................................................. 448
SYNCHRONIZE STAMPS ................................................................. 449
TUNE INDEX ....................................................................................... 449
UNLOCK ............................................................................................... 450
UNLOAD .............................................................................................. 450
UPDATE STATISTICS ....................................................................... 451
VALIDATE ........................................................................................... 451
Utility Programs (CMS Commands) ................................................................. 452
IDMSDBAN ........................................................................................................ 452
IDMSDIRL ........................................................................................................ 454
IDMSLOOK ....................................................................................................... 455
IDMSRPTS ....................................................................................................... 455
IDMSRSTC ....................................................................................................... 456
IDMSRSTT ....................................................................................................... 457

FASTLOAD Format Program Sample Listing .............................................. 459

IDMSRSTT Macro Statements ................................................................. 478
IDMSRSTT BUFSIZE .................................................................................. 479
Syntax ............................................................................................................ 479
Parameter ..................................................................................................... 479
Usage ............................................................................................................ 480
IDMSRSTT RECNAME .................................................................................. 480
Syntax ............................................................................................................ 480
Parameters ................................................................................................... 481
Usage ............................................................................................................ 482
IDMSRSTT SETPTR ...................................................................................... 483
Syntax ............................................................................................................ 483
Parameters ................................................................................................... 483
Usage ............................................................................................................ 484
IDMSRSTT FIELD ....................................................................................... 484
Syntax ............................................................................................................ 485
Parameters ................................................................................................... 485
Usage ............................................................................................................ 486
IDMSRSTT END ............................................................................................ 488
Syntax ............................................................................................................ 488
END (IDMSRSTT) ....................................................................................... 488

Common Facilities for Distributed Transactions ...................................... 489
Reporting on Distributed Transactions ........................................................ 489
Manual Recovery Input Control File ......................................................... 491
Manual Recovery Output Control File ....................................................... 492
Execution JCL Changes ............................................................................. 493
Utilities

You submit utility statements and programs to CA IDMS/DB to request the following types of operations:

- Backup and recovery
- CA IDMS system log maintenance
- Database area maintenance
- Database loading and restructuring
- Database integrity checking
- Database reporting
- Enhancing SQL data access
- Load module management

For more information, see the following topics:

- Utility Operations (see page 19)
- Command Facility Considerations (see page 26)
- Utility Statements (see page 28)
- Utility Programs (CA IDMS/DB Utilities) (see page 300)
- z/OS JCL - IDMSBCF and Utilities (see page 355)
- z/VSE JCL (see page 389)
- CMS Commands - IDMSBCF and Utilities (see page 428)
- FASTLOAD Format Program Sample Listing (see page 459)
- IDMSRSTT Macro Statements (see page 478)
- Common Facilities for Distributed Transactions (see page 489)
Utility Operations

The following utility statements and programs are presented by database and system operation functions.

- Using the Utility Statements (see page 19)
  - Utility Descriptions (see page 19)
  - SQL and Non-SQL (see page 22)

Using the Utility Statements

Utility Descriptions

Backup and recovery utilities

CA IDMS/DB provides the following utilities for backup and recovery operations:

<table>
<thead>
<tr>
<th>Utility</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARCHIVE</td>
<td>Offload disk journal files to archive files</td>
</tr>
<tr>
<td>JOURNAL</td>
<td></td>
</tr>
<tr>
<td>BACKUP</td>
<td>Back up database areas</td>
</tr>
<tr>
<td>EXTRACT</td>
<td>Extract AFTR images from archived journal file and write them to an extract file; the extract file can be used as input to the ROLLFORWARD utility</td>
</tr>
<tr>
<td>JOURNAL</td>
<td></td>
</tr>
<tr>
<td>FIX</td>
<td>Rewrite a tape journal file</td>
</tr>
<tr>
<td>ARCHIVE</td>
<td></td>
</tr>
<tr>
<td>PAGE</td>
<td>Verify or modify the contents of a database page</td>
</tr>
<tr>
<td>MERGE</td>
<td>Merge archived journal files of data sharing group members. The output file can be used as input to the ROLLFORWARD, ROLLBACK, EXTRACT JOURNAL, and MERGE ARCHIVE utility statements.</td>
</tr>
<tr>
<td>PRINT</td>
<td>Report on transaction activity</td>
</tr>
<tr>
<td>JOURNAL</td>
<td></td>
</tr>
<tr>
<td>RESTORE</td>
<td>Restore backed up database areas</td>
</tr>
<tr>
<td>ROLLBACK</td>
<td>Restore files or areas to earlier states using journal information</td>
</tr>
</tbody>
</table>
Utility | Purpose
--- | ---
ROLLFO | Update a backup copy of a file or area using journal information
RWARD |
UNLOCK | Remove locks from an area

⚠️ Note: For more information about backup and recovery operations, see the sections on the individual utilities in this document or see the *CA IDMS Database Administration Section*.

Log maintenance utilities

CA IDMS/DB provides the following utilities for maintaining the DC/UCF system log:

<table>
<thead>
<tr>
<th>Utility</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARCHIVE LOG</td>
<td>Offload system log to archive file</td>
</tr>
<tr>
<td>ARCHIVE TRACE</td>
<td>Offloads the contents of the DC/UCF trace area to an archive file.</td>
</tr>
<tr>
<td>PRINT LOG</td>
<td>Print all or part of a system log or archive log</td>
</tr>
</tbody>
</table>

⚠️ Note: For more information about maintaining the system log, see the sections on the individual utilities in this document or see the *CA IDMS System Operations Section*.

Area maintenance utilities

CA IDMS/DB provides the following utilities for database and journal file maintenance:

<table>
<thead>
<tr>
<th>Utility</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLEANUP</td>
<td>Erase logically deleted records</td>
</tr>
<tr>
<td>EXPAND PAGE</td>
<td>Increase page size for a database file</td>
</tr>
<tr>
<td>FIX PAGE</td>
<td>Verify or modify the contents of a database page</td>
</tr>
<tr>
<td>FORMAT</td>
<td>Prepare a file, area, or segment for use by CA IDMS/DB</td>
</tr>
<tr>
<td>INSTALL STAMPS</td>
<td>Store synchronization stamps for an SQL-defined database</td>
</tr>
<tr>
<td>LOCK</td>
<td>Lock an area or segment in a batch job</td>
</tr>
<tr>
<td>PRINT PAGE</td>
<td>Print the contents of database pages</td>
</tr>
<tr>
<td>PRINT SPACE</td>
<td>Report on space utilization in areas</td>
</tr>
<tr>
<td>SYNCHRONIZE STAMPS</td>
<td>Display the stamps in the catalog and the data area(s) and update the stamps</td>
</tr>
<tr>
<td>TUNE INDEX</td>
<td>Walk a sorted index to cause the adoption of orphaned index records</td>
</tr>
<tr>
<td>UNLOCK</td>
<td>Remove locks from an area</td>
</tr>
</tbody>
</table>
Utility | Purpose
--- | ---
UPDATE STATISTICS | Update statistics used by CA IDMS/DB to optimize access to an SQL-defined database

**Note:** For more information about database area maintenance, see the sections on the individual utilities in this document or see the *CA IDMS Database Administration Section.*

### Database loading and restructuring utilities

CA IDMS/DB provides the following utilities for loading and restructuring a database:

<table>
<thead>
<tr>
<th>Utility</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUILD</td>
<td>Build or rebuild indexes and build referential constraints for an SQL-defined database</td>
</tr>
<tr>
<td>CONVERT CATALOG</td>
<td>Convert the SYSTEM schema within a catalog to reflect the definition specified by the SYSTEM schema DDL delivered with the current release of CA IDMS</td>
</tr>
<tr>
<td>CONVERT PAGE</td>
<td>Change the page range for an area or change the maximum number of records that can be stored on a page of an area</td>
</tr>
<tr>
<td>CREATE DSMODEL</td>
<td>Create a temporary model of data set attributes to use for dynamic file allocation in conjunction with the REORG utility</td>
</tr>
<tr>
<td>FASTLOAD</td>
<td>Load data into a non-SQL-defined database for the first time</td>
</tr>
<tr>
<td>IDMSDIRL</td>
<td>Load the IDMSNTWK version 1 schema and the IDMSNWKA subschema into a data dictionary</td>
</tr>
<tr>
<td>IDMSRSTC</td>
<td>Generate IDMSRSTT macro statements for restructuring a non-SQL-defined database</td>
</tr>
<tr>
<td>LOAD</td>
<td>Load data into an SQL-defined database</td>
</tr>
<tr>
<td>MAINTAIN INDEX</td>
<td>Build, rebuild, or delete indexes in a non-SQL-defined database</td>
</tr>
<tr>
<td>RELOAD</td>
<td>Reload a database unloaded by UNLOAD</td>
</tr>
<tr>
<td>REORG</td>
<td>Unload and reload all or part of a database</td>
</tr>
<tr>
<td>RESTRUCTURE CONNECT</td>
<td>Connect new prior and owner pointers in existing sets in a non-SQL-defined database</td>
</tr>
<tr>
<td>RESTRUCTURE E</td>
<td>Modify record occurrences to match new schema specifications</td>
</tr>
<tr>
<td>UNLOAD</td>
<td>Unload all or part of a database</td>
</tr>
<tr>
<td>VALIDATE</td>
<td>Check referential constraints for an SQL-defined database</td>
</tr>
</tbody>
</table>

**Note:** For more information about loading data into the database, see the sections on the individual utilities in this document or see the *CA IDMS Database Administration Section.*
Integrity checking utilities

CA IDMS/DB provides the following utilities for checking database integrity:

<table>
<thead>
<tr>
<th>Utility</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDMSDBAN</td>
<td>Analyze the structure of an existing non-SQL-defined database</td>
</tr>
<tr>
<td>PRINT INDEX</td>
<td>Report on system owned indexes and indexed sets</td>
</tr>
<tr>
<td>VALIDATE</td>
<td>Check referential constraints for an SQL-defined database</td>
</tr>
</tbody>
</table>

Note: For more information about checking database integrity, see the sections on the individual utilities in this document or see the CA IDMS Database Administration Section.

Database reporting utilities

CA IDMS/DB provides the following utilities for reporting on database structure and contents:

<table>
<thead>
<tr>
<th>Utility</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDMSDBAN</td>
<td>Analyze the structure of an existing non-SQL-defined database</td>
</tr>
<tr>
<td>IDMSLOOK</td>
<td>Report on the contents of load modules</td>
</tr>
<tr>
<td>IDMSRPT</td>
<td>Report on information stored in a data dictionary</td>
</tr>
<tr>
<td>PRINT INDEX</td>
<td>Report on system owned indexes and indexed sets</td>
</tr>
<tr>
<td>PRINT JOURNAL</td>
<td>Report on transaction activity</td>
</tr>
<tr>
<td>PRINT PAGE</td>
<td>Print the contents of database pages</td>
</tr>
<tr>
<td>PRINT SPACE</td>
<td>Report on space utilization in areas</td>
</tr>
</tbody>
</table>

SQL and Non-SQL

Some utilities can only be used for SQL or non-SQL databases. Others can be used without regard for the type of database you are using.

SQL database only

- BUILD
- INSTALL STAMPS
- LOAD
- SYNCHRONIZE STAMPS
- VALIDATE

Non-SQL database only
- CLEANUP
- FASTLOAD
- MAINTAIN INDEX
- RESTRUCTURE CONNECT
- RESTRUCTURE
- IDMSRSTC

For both SQL and non-SQL databases

- ARCHIVE JOURNAL
- ARCHIVE LOG
- ARCHIVE TRACE
- BACKUP
- CONVERT CATALOG
- CONVERT PAGE
- CREATE DSMODEL
- EXPAND PAGE
- EXTRACT JOURNAL
- FIX ARCHIVE
- FIX PAGE
- FORMAT
- LOCK
- MERGE ARCHIVE
- PRINT INDEX
- PRINT JOURNAL
- PRINT LOG
- PRINT PAGE
- PRINT SPACE
- PUNCH
Central Version Considerations

The following considerations apply when executing a utility under a central version through the online or batch command facility.

Area usage mode

To execute a utility under a central version, the affected areas must be available to the central version in the appropriate mode. For utilities that perform updates, the affected areas must be in update mode to the central version. For utilities that perform only retrievals, the affected areas must be in retrieval or update mode. If the previous requirement is not met, you receive a DB002352 error message indicating that the required lock mode is not available.

Committing prior work

Before executing certain utilities through central version, you must commit any previous work that has been done within the current session. This requirement applies to the following utilities:

- FIX PAGE
- FORMAT AREA
• FORMAT SEGMENT

• LOCK AREA

• LOCK SEGMENT

The following sequence of statements illustrates how to commit prior work before issuing a FORMAT AREA statement:

SELECT * FROM SYSTEM.TABLE;
COMMIT;
FORMAT AREA VSAMT.KSDS2;

If you omit the COMMIT, you receive a DB002043 error message: Command not allowed with an open transaction.

Log messages

If you run a FORMAT statement or FIX PAGE statement under central version, an informational message is written to the log identifying the area name being updated and the time of the update.

Batch-only utilities

If you attempt to execute a utility under central version that is supported only in batch local mode, such as UNLOCK or FORMAT FILE, you receive a DB002990 error message indicating that the statement is not supported in central version.
Command Facility Considerations

Statement components

Utility statements consist of:

- **Keywords** that:
  - Identify the action requested by the statement (for example, BACKUP or PRINT SPACE)
  - Specify the type of entity (for example, AREA or SEGMENT) that is the object of the requested action
  - Place qualifications on the requested action, either by themselves (for example, SHARE or NO REPORT) or in conjunction with user-supplied values (for example, START AT 1999-12-08-06.00.00)

- **User-supplied values** that:
  - Identify specific occurrences of entities (for example, the area EMP_SPACE or the database segment DEMOSEG)
  - Specify data values (for example, 983 or 'Boston')

- **Separators** that separate keywords and user-supplied values from one another. A separator can be a space, a comment, or the end of a line.

Separators are not required:

- Before or after a value in single quotation marks
- A comma (,)
- An equal sign (=)
- Left and right parentheses ()
- A period (.)
- A semicolon (;)

Delimiting statements

When you use the command facility to submit utility statements, you must terminate each statement with a statement delimiter, which by default is a semicolon (;). You can enter the delimiter either on the same line as the rest of the statement or on a separate line. For example, the two statements shown next are equivalent:

```
format area emp-region-area;
format area emp-region-area;
```
Continuing statements

You can code utility statements on one or more lines. No special character is required to indicate that a statement continues on the next line.

Quotation marks around identifiers

In a utility statement, you must enclose a database entity identifier in double quotation marks if the identifier includes significant lowercase characters, special characters (except a dash (-)), or blanks. Place the quotation marks only around the individual identifier for which they are required (for example, SALESSEG."EST%_SPACE").

Note that all identifiers can contain dashes even when not quoted. For example, SALES-SEG."EST%_SPACE".

More Information

- For quotation marks with identifiers, see SQL Reference (https://docops.ca.com/display/IDMSCU19/SQL+Reference).
- For database entity identifiers, see Administrating CA IDMS Database (https://docops.ca.com/display/IDMS19/Administrating+CA+IDMS+Database).

Maximum statement length

A utility statement can be at most 8,192 bytes long. If you are using only single-byte characters, the maximum number of characters equals the maximum number of bytes. If any user-supplied values contain double-byte characters, the maximum number of characters is less than the maximum number of bytes.
Utility Statements

This section contains the following utility statements:

- ARCHIVE JOURNAL (see page 29)
- ARCHIVE LOG (see page 34)
- ARCHIVE TRACE (see page 37)
- BACKUP (see page 38)
- BUILD (see page 43)
- CLEANUP (see page 49)
- CONVERT CATALOG (see page 52)
- CONVERT PAGE (see page 53)
- CREATE DSMODEL (see page 61)
- EXPAND PAGE (see page 64)
- EXTRACT JOURNAL (see page 67)
- FASTLOAD (see page 75)
- FIX ARCHIVE (see page 87)
- FIX PAGE (see page 91)
- FORMAT (see page 95)
- INSTALL STAMPS (see page 106)
- LOAD (see page 109)
- LOCK (see page 123)
- MAINTAIN INDEX (see page 125)
- MERGE ARCHIVE (see page 141)
- PRINT INDEX (see page 146)
- PRINT JOURNAL (see page 158)
- PRINT LOG (see page 162)
- PRINT PAGE (see page 168)
- PRINT SPACE (see page 174)
- PRINT TRACE (see page 179)
- PUNCH (see page 185)
- RELOAD (see page 187)
- REORG (see page 200)
- RESTORE (see page 237)
- RESTRUCTURE (see page 241)
- RESTRUCTURE CONNECT (see page 248)
- ROLLBACK Utility (see page 252)
- ROLLFORWARD (see page 260)
- SYNCHRONIZE STAMPS (see page 269)
- TUNE INDEX (see page 272)
- UNLOAD (see page 279)
- UNLOCK (see page 288)
The ARCHIVE JOURNAL utility offloads, to one or more archive journal files, the entries in one or more disk journal files.

The options you can choose depend upon the type of recovery being performed and whether or not the DC/UCF system is active.

This article describes the following information:

- Authorization (see page 29)
- Syntax (see page 29)
- Parameters (see page 30)
- Usage (see page 31)
- JCL Considerations (see page 33)
- Example (see page 33)
- Sample Output (see page 33)

Authorization

<table>
<thead>
<tr>
<th>To</th>
<th>You need this privilege</th>
<th>On</th>
</tr>
</thead>
<tbody>
<tr>
<td>Archive one or more journal files</td>
<td>USE</td>
<td>The DMCL</td>
</tr>
</tbody>
</table>

Syntax

```
ARCHIVE JOURNAL
BUFFERS buffer-count
ALL
AUTO
AUTDALL
FULL
READ journal-file-name
REPORT
RESTART journal-file-name
```
Parameters

- **BUFFERS**
  Specifies the number of buffer pages to be used during condense processing.
  Condense processing involves copying the before image of unfinished transactions back to the disk journal file after it is offloaded. Condense processing of journal segments takes place when both of the following apply:
  
  - You specify **AUTO** and
  - The DC/UCF system is active or after an abnormal system termination

  The buffer is used for storing before images of unfinished transactions before copying the images back to the disk journal file.
  The buffer page size is equal to the block size of the disk journal file.

- **buffer-count**
  An integer in the range 2 through 32,767. The default is 5.

- **ALL**
  Specifies that all non-empty disk journal files are to be offloaded, starting with the file containing the oldest entries.

  **Note:** This option is not allowed while the DC/UCF system is active or after an abnormal system shutdown, unless you are beginning a section recovery of the entire system.

  After offloading the journal files, they are marked as empty.
  Before images of unfinished transactions are not rewritten or condensed after offloading the journal files. These before images are needed if you need to recover from an abnormal shutdown.

- **AUTO**
  Directs the ARCHIVE JOURNAL utility to select a single file to offload:
  
  - While the DC/UCF system is active or after an abnormal system termination, the oldest full disk journal file is selected.
  - After offloading the file, it is condensed.
  - After condensing the selected file, a new, empty journal segment is created.
  - If no full disk journal file exists, no file is offloaded or condensed.

  - After a normal DC/UCF system shutdown, the oldest non-empty disk journal file is selected.
  - After offloading the file, it is marked as empty.
AUTOALL

- While the DC/UCF system is active or after an abnormal system shutdown, the oldest full disk journal file is offloaded. The file is condensed after it is offloaded and a new empty journal segment is created.

- After a normal DC/UCF shutdown, specifies that all non-empty disk journal files are to be offloaded, starting with the file containing the oldest entries. After offloading the journal files, they are marked as empty.

FULL

Directs the ARCHIVE JOURNAL utility to offload all full disk journal files associated with the database, starting with the file containing the oldest entries. After offloading each file, it is marked as empty.

Before images of unfinished transactions are not condensed and rewritten. These before images are needed if you need to recover from an abnormal shutdown.

Note: This option is not allowed while the DC/UCF system is active or after an abnormal system shutdown, unless you are beginning a section recovery of the entire system.

READ

Directs the ARCHIVE JOURNAL utility to offload a single disk journal file without condensing it or marking it as empty afterwards.

If you do not specify a file name, the oldest non-empty disk journal file is offloaded.

journal-file-name

The name of the disk journal file to be read.

REPORT

Displays information about all disk journal files. No archiving or condensing of the journal files is performed.

RESTART

Directs the ARCHIVE JOURNAL utility to restart an archive journal operation that terminated abnormally. If the operation failed:

- While a disk journal file was being offloaded, the offload operation is restarted.

- While a disk journal file was being condensed, the condense operation is restarted.

journal-file-name

The name of the disk journal file being offloaded at the time of the abend.

Usage

Summary of offload parameter options
In summary, if the DC/UCF system is not active, one or more disk journals may be processed, and after a file is offloaded, it is marked empty.

If the DC/UCF system is active, only the oldest full disk journal file is processed (as if the AUTO option is specified). After the file has been offloaded it is condensed and a new journal segment is created.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>While DC/UCF Active or After Abnormal Termination</th>
<th>After Normal Shutdown</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>Not Allowed</td>
<td>All non-empty disk journal files, beginning with the oldest file, are offloaded. Offloaded files are marked as empty.</td>
</tr>
<tr>
<td>AUTOALL</td>
<td>Oldest full disk journal file is offloaded. Condenses offloaded file, then creates a new, empty journal segment.</td>
<td>All non-empty disk journal files, beginning with the oldest file, are offloaded. Offloaded files are marked as empty.</td>
</tr>
<tr>
<td>AUTO</td>
<td>Oldest full disk journal file is offloaded. Condenses offloaded file, then creates a new, empty journal segment.</td>
<td>Oldest non-empty disk journal file is offloaded and marked as empty.</td>
</tr>
<tr>
<td>FULL</td>
<td>Not Allowed</td>
<td>All full disk journal files, beginning with the oldest file, are offloaded.</td>
</tr>
</tbody>
</table>

How to submit the ARCHIVE JOURNAL statement

You submit the ARCHIVE JOURNAL statement to CA IDMS/DB only through the batch command facility. You must run the batch command facility in local mode.

You normally specify AUTO

Normally, you archive journal files as they become full. For this purpose, specify AUTO.

The other options are used only in special circumstances, such as recovering damaged files.

Use with two-phase commit feature

ARCHIVE JOURNAL automatically preserves journal records for incomplete or unforgotten distributed transaction records when condensing a journal file.

Obtaining a status report

ARCHIVE JOURNAL always produces a report showing the status of disk journal files. Specify the REPORT option if you only want to obtain the report and you do not want to archive or condense journal information.

Use with change tracking

If the CV whose journals are being archived uses change tracking to record changes to its database environment, the JCL used to archive its journal files should reference the CV's SYSTRK files and should not include file assignments for the journal files. This ensures that any changes made to the journal files in use by the CV are known to the ARCHIVE JOURNAL utility so that it operates on the correct set of files.
Performance consideration

A reduction in run-time for the ARCHIVE JOURNAL utility might occur if you process the journals using the QSAM access method. To invoke QSAM against the journals specify QSAMAREA=ARCHIVE. JOURNAL in the SYSIDMS file used by the ARCHIVE JOURNAL utility.

JCL Considerations

When you submit an ARCHIVE JOURNAL statement through the batch command facility, the JCL to execute the facility must include statements to define:

- The disk journal files or the CV's SYSTRK files if change tracking is in use by the CV
- The archive journal files.

Example

While your DC/UCF system is active, you can initiate a job that will offload full disk journal files. To do this, specify the AUTO option of the ARCHIVE JOURNAL statement:

```
archive journal auto;
```

Sample Output

After successful completion of the ARCHIVE JOURNAL statement submitted with the AUTO option, the CA IDMS Batch Command Facility produces the following listing:

```
IDMSBCF mm/dd/yy PAGE 1
IDMS Batch Command Facility
ARCHIVE JOURNAL ;

JOURNAL DISK FILES STATUS REPORT
FILENAME   SEGMENT LORBN HIRBN FULL ACTIVE STATUS CV ACTIVE
SYSJRNL2   77      10   1000  NO  YES  NON-AJNL  YES
75      8      9
SYSJRNL1   76      10   1000  YES NO   NON-AJNL  YES
```
ARCHIVE LOG

The ARCHIVE LOG utility offloads the contents of the DC/UCF system log to an archive file.

This article describes the following information:

- Authorization (see page 35)
- Syntax (see page 35)
- Parameters (see page 35)
- Usage (see page 35)
- JCL Considerations (see page 36)
- Example (see page 36)
- Sample Output (see page 36)
Authorization

To archive a system log you need DBAWRITE privileges in the SYSTEM.DDLCLOG area in the dictionary associated with the DC/UCF system whose log you want to archive.

Syntax

```
ARCHIVE LOG [PRINT [UTC]]
```

Parameters

- **PRINT**
  Prints a copy of the contents of the archived log. If you do not specify PRINT, a copy of the archived log is not printed.

- **UTC**
  Displays the time stamps of each print line as UTC time. If not specified, times are displayed in local time.

Usage

**How to submit the ARCHIVE LOG statement**

You submit an ARCHIVE LOG statement only through the batch command facility. You must be running CA IDMS/DB in local mode, without journaling.

**When to use ARCHIVE LOG**

Use the ARCHIVE LOG utility only when the system log is being written to the DDLCLOG area.

**When not to use ARCHIVE LOG**

If the system log is assigned to one or two sequential files, you should use the appropriate operating system utility (for example, IEBGENER for z/OS systems or DITTO for z/VSE systems) to archive the contents of the log file.

**Archiving the log for an active system**

When you submit an ARCHIVE LOG statement while the DC/UCF system is active, CA IDMS/DB archives the contents of the DDLCLOG area up to, but not including, the page to which the system is currently writing.
JCL Considerations

When you submit an ARCHIVE LOG statement through the batch command facility, the JCL to execute the facility must include statements to define the following:

- Log area (DDLLCLOG)
- Message area (DDLLCMSG)
- Dummied journal file
- Archive log file being created

⚠️ Note: For more information about the generic JCL used to execute the batch command facility, see the section for your operating system in this section.

Example

The following ARCHIVE LOG statement requests that the contents of the DC/UCF system log be offloaded to an archive file and to print the contents of the archived log.

archive log print;

Sample Output

The ARCHIVE LOG utility produces the following standard listing:

```
IDMSBCF mm/dd/77 PAGE 1
ARCHIVE LOG;
CA IDMS-DB/DC Print Log Utility
/DD is a Proprietary Software Product
volser Licensed from CA
/yy hh:mm:ss 1
*** PAGE 000030002 STATUS 0000 mmddyy 19.14.59 6 $
*** PAGE 000031001 STATUS 0000 mmddyy 14.20.00 6 $
*** PAGE 000031500 STATUS 0000 mmddyy 10.19.58 6 $
*** PAGE 000031750 STATUS 0000 mmddyy 14.30.01 6 $
*** PAGE 000031875 STATUS 0000 mmddyy 16.46.34 6 $
*** PAGE 000031937 STATUS 0000 mmddyy 17.54.59 6 $
*** PAGE 000031968 STATUS 0000 mmddyy 18.34.58 6 $
*** PAGE 000031984 STATUS 0000 mmddyy 18.54.59 6 $
*** PAGE 000031992 STATUS 0000 mmddyy 19.04.58 6 $
*** PAGE 000031996 STATUS 0000 mmddyy 19.10.00 6 $
*** PAGE 000031998 STATUS 0000 mmddyy 19.14.58 6 $
*** PAGE 000031999 STATUS 0000 mmddyy 19.14.59 6 $
*** PAGE 000032000 STATUS 0000 mmddyy 19.14.59 6 $
*** PAGE 000030002 STATUS 0000 mmddyy 19.14.59 6 $
*** PAGE 000032000 STATUS 0000 mmddyy 19.14.59 6 $
*** PAGE 000031001 STATUS 0000 mmddyy 14.20.00 6 $
*** PAGE 000031500 STATUS 0000 mmddyy 10.19.58 6 $
```

16-Jan-2018
The ARCHIVE TRACE utility offloads the contents of the DC/UCF trace area to an archive file.

Authorization

To archive trace information you need DBAWRITE privileges in the SYSTEM.DDLDCTRC area.

Syntax

The following diagram shows the syntax for the ARCHIVE TRACE utility statement:

```
ARCHIVE TRACE [PRINT [UTC]]
```
Parameters

This section describes the parameters for the ARCHIVE TRACE utility:

- **PRINT**
  Prints a copy of the contents of the archived trace information. If you do not specify PRINT, a copy of the archived trace information is not printed.

  **Note:** For a description of the output produced by the PRINT option of the ARCHIVE TRACE utility, see the PRINT TRACE Utility (see page ).

- **UTC**
  Displays the time stamps of each print line as UTC time. If not specified, times are displayed in local time.

Example

The following example writes the contents of a system trace area (DDLDCTRC) to an archive file and empties the area.

```
ARCHIVE TRACE
```

BACKUP

The BACKUP utility copies one or more areas in a database to a backup file. The backup file can be used later as input for a restore operation.

**Note:** The format of files produced by the BACKUP utility is not compatible with the format of backup files produced by the 10.2 IDMSDUMP utility program. Files produced by BACKUP can only be used by the RESTORE utility and files produced by IDMSDUMP can only be used with IDMSRSTR.

This article describes the following information:

- Authorization (see page 39)
- Syntax (see page 39)
- Parameters (see page 39)
- Usage (see page 40)
- JCL Considerations (see page 41)
- Examples (see page 41)
Authorization

<table>
<thead>
<tr>
<th>To</th>
<th>You Need This Privilege</th>
<th>On</th>
</tr>
</thead>
<tbody>
<tr>
<td>Back up an area</td>
<td>DBAREAD</td>
<td>The area</td>
</tr>
<tr>
<td>Back up a file</td>
<td>DBAREAD</td>
<td>The area(s) to which the file maps</td>
</tr>
</tbody>
</table>

Syntax

```
BACKUP AREA segment-name.area-name [SHARE]
```

```
FILE segment-name.file-name [SHARE]
```

```
SEGMENT segment-name [SHARE]
```

Parameters

- **AREA**
  Directs the BACKUP utility to back up one or more areas. Multiple area names must be separated by commas.

- **segment-name**
  The name of the segment associated with an area to be backed up.

- **area-name**
  The name of an area.

- **SHARE**
  Specifies that no locks are to be placed on the named areas. When you specify SHARE, each specified area is backed up regardless of whether a lock has been placed on the area by another program. SHARE allows the named areas to be backed up while another job is updating those areas at the same time, for example, it needs to be specified when taking a "hot backup" during which the areas that are being backed up are being updated by transactions executing under the central version.

  By default, if you do not specify SHARE, an external lock is placed on each specified area for the duration of the backup operation. If an external lock cannot be placed on an area, that area will not be backed up, and the backup operation will terminate with an error. Therefore, no more areas will be backed up. An external lock means that it is placed physically using the normal SMP lock.
**CA IDMS Reference - 19.0**

**Note:** When you specify SHARE and *do not* vary the affected areas for retrieval only, the copy of the database created by the BACKUP utility may not be usable for restore operations.

- **FILE**
  Directs the BACKUP utility to back up one or more files. Multiple file names must be separated by commas.

- **segment-name**
  The name of a segment associated with a file to be backed up.

- **file-name**
  The name of a file.

- **SEGMENT segment-name**
  The name of the segment to be backed up.

- **BY AREA**
  Specifies that each area defined within the segment is to be backed up. AREA is the default.

- **BY FILE**
  Specifies that each file within the segment is to be backed up.

**Note:** The SHARE option is only valid for area processing. When you specify the BACKUP SEGMENT command with the BY FILE option, the SHARE option is ignored. If you specify the BY FILE option with the BACKUP SEGMENT command and the SHARE option is omitted, area locks are not set.

### Usage

**How to submit the BACKUP statement**

You submit the BACKUP statement only through the batch command facility. You must run the batch command facility in local mode.

**When to use SHARE**

You can specify SHARE and get a backup usable for restore operations if you vary the affected area(s) for retrieval only. This prevents other users from changing the contents of the area during the backup operation.

If you specify SHARE without varying the area(s) for retrieval only, you should exercise extreme caution when making use of the backup.

**BACKUP by file does not lock areas**
When you back up by file, the BACKUP utility does not lock the associated area(s). Therefore, to preserve the integrity of the area, you should vary affected areas for retrieval only.

**RESTORE the same object you backed up**

If you back up by area, restore by area. If you back up by file, restore by file.

**Number of BACKUP statements per BCF-job**

Only one BACKUP statement per BCF-job is allowed. Specifying multiple BACKUP statements for the same BCF-job will result in all BACKUP files, except for the last one, being overwritten. This is caused by the way syntax parsing has been implemented, for example, for each BACKUP statement the BACKUP utility (such as, module IDMSUBKP) is called that at the beginning will open the BACKUP file specified by <ddname/filename/linkname> SYS001 in OUTPUT mode and will close it before returning to the syntax parsing module.

**JCL Considerations**

When you submit a BACKUP statement through the batch command facility, the JCL to execute the facility must include statements to define:

- The files associated with the areas to be backed up
- The archive file which will contain the backup

⚠️ **Note:** For more information about the generic JCL used to execute the batch command facility, see the section for your operating system in this section.

**Examples**

**Back up by area**

The following example directs the BACKUP utility to back up three database areas.

```plaintext
backup area empdemo.emp-demo-region,
        empdemo.org-demo-region,
        empdemo.ins-demo-region;
```

**Back up by file**

The following example directs the BACKUP utility to back up three database files.

```plaintext
backup file empdemo.empdemo,
         empdemo.orgdemo,
         empdemo.insdemo;
```

**Back up by segment**
The following example directs the BACKUP utility to back up all areas in the empdemo segment.

```
backup segment empdemo;
```

**Sample Output**

**Back up by area**

When the backup by area operation in the previous example is completed, the BACKUP utility provides the following report.

```
IDMSBCF nn.  CA IDMS Batch Command Facility
             mm/dd/yy    PAGE 1

BACKUP AREA EMPDEMO.EMP-DEMO-REGION,
          EMPDEMO.ORG-DEMO-REGION,
          EMPDEMO.INS-DEMO-REGION;

UT015006 BACKUP file created on yyyy-mm-dd-hh.mm.ss.ffffff
UT015005 Max Archive record size is 4,280
UT000038 Starting BACKUP of area EMPDEMO.EMP-DEMO-REGION
UT000040 BACKUP complete
UT000038 Starting BACKUP of area EMPDEMO.ORG-DEMO-REGION
UT000040 BACKUP complete
UT000038 Starting BACKUP of area EMPDEMO.INS-DEMO-REGION
UT000040 BACKUP complete

Status = 0        SQLSTATE = 00000
```

**Output from back up by file**

When the backup by file operation in the previous example is completed, the BACKUP utility provides the following report.

```
IDMSBCF nn.  CA IDMS Batch Command Facility
             mm/dd/yy    PAGE 3

BACKUP FILE EMPDEMO.EMPDEMO,
          EMPDEMO.ORGDEMO,
          EMPDEMO.INSDEMO;

UT015006 BACKUP file created on yyyy-mm-dd-hh.mm.ss.ffffff
UT015005 Max Archive record size is 4,280
UT000039 Starting BACKUP of file EMPDEMO.EMPDEMO
UT000040 BACKUP complete
UT000039 Starting BACKUP of file EMPDEMO.ORGDEMO
UT000040 BACKUP complete
UT000039 Starting BACKUP of file EMPDEMO.INSDEMO
UT000040 BACKUP complete

Status = 0        SQLSTATE = 00000
```

**Output from backup by segment**

When the backup by segment operation in the previous example is completed, the BACKUP utility provides the following report.

```
IDMSBCF nn.  CA IDMS Batch Command Facility
             mm/dd/yy    PAGE 5

```

16-Jan-2018
BUILD

The BUILD utility statement builds indexes and referential constraints linked through an index on tables that are being loaded with a phased or stepped LOAD.

The BUILD utility can also be used to reorganize existing indexes.

The BUILD utility works only on tables in an SQL-defined database.

<table>
<thead>
<tr>
<th>Type of BUILD</th>
<th>What it does</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete BUILD</td>
<td>Runs all four steps</td>
</tr>
<tr>
<td>Stepped BUILD</td>
<td>Runs one step at a time with intermediate file sorting required between each step</td>
</tr>
</tbody>
</table>

This article describes the following information:

- Authorization (see page 44)
- Syntax (see page 44)
- Parameters (see page 45)
- Usage (see page 47)
- JCL Considerations (see page 47)
- Examples (see page 48)
- Sample Output (see page 49)

More Information

- For more information about using the SHARE option, see Administrating CA IDMS Database (https://docops.ca.com/display/IDMS19/Administrating+CA+IDMS+Database).
- For more information about varying areas, see CA IDMS System Tasks and Operator Commands (https://docops.ca.com/display/IDMSCU19/CA+IDMS+System+Tasks+and+Operator+Commands).
Authorization

<table>
<thead>
<tr>
<th>To</th>
<th>You need this privilege</th>
<th>On</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build indexes and/or referential constraints on a table</td>
<td>INSERT</td>
<td>The indexed or referencing table</td>
</tr>
</tbody>
</table>

For more information about designing indexes and referential constraints in SQL-defined databases, see the Database Design Section.

For more information about defining indexes and referential constraints in SQL-defined databases, see the Database Administration Section.

Syntax

Syntax for complete BUILD

```
BUILD INDEXES ALL FOR schema-name.table-id
```

NO VALIDATE

```
NOTIFY notify-record-count
```

```
BUILD ALL ERRORS error-count CONTINUE
```

Syntax for stepped BUILD

```
BUILD
STEP1 INDEXES CONSTRAINTS BOTH FOR schema-name.table-id
STEP2
STEP3
STEP4
```

```
NOTIFY notify-record-count
```

```
BUILD ALL ERRORS error-count CONTINUE
```
Parameters

- **INDEXES**
  Directs the BUILD utility to build the indexes only.
  By default, if you do not specify what is to be built, both indexes and constraints are built.

- **ALL**
  Directs the BUILD utility to build all indexes (clustered and non-clustered). If ALL is not specified, only non-clustered indexes will be built.

- **CONSTRAINTS**
  Directs the BUILD utility to build referential constraints only.
  By default, if you do not specify what is to be built, both indexes and constraints are built.

- **BOTH**
  Directs the BUILD utility to build both indexes and relationships. BOTH is the default.

- **FOR**
  Specifies the table for which indexes and/or constraints are to be built.

  - **schema-name.**
    The name of the schema that defines the table.

  - **table-id**
    The identifier of the table.

- **NO VALIDATE**
  Directs the BUILD utility not to validate referential constraints.
  If you specify NO VALIDATE, you will have to execute the VALIDATE utility before you can use the table(s).
  By default, the validation is performed.
  You can specify NO VALIDATE only for a complete BUILD.

- **NOTIFY**
  Directs the BUILD utility to send a message to the operator whenever a specified number of records are processed.
  The message states the phase and step currently being executed and the number of records that have been processed.

- **notify-record-count**
  The number of records to process before sending a message.
• **BUILD ERRORS**  
When errors are detected, directs the BUILD utility to either continue processing or stop after a specified number of errors are detected.  
By default, processing is stopped after the first error is detected.  
Detected errors are listed in the report generated by the BUILD utility.

• **VALIDATE ERRORS**  
When errors are detected in the validation process, directs the BUILD utility to either continue processing or stop processing after a specified number of errors are encountered.  
By default, processing is stopped after the first error is detected.  
Detected errors are listed in the report generated by the BUILD utility.  
You can specify VALIDATE ERRORS only for a complete BUILD.

• **ALL ERRORS**  
Directs the BUILD utility to either continue when any errors are detected, or stop after a specified number of errors are detected.  
By default, processing is stopped after the first error is detected.  
Detected errors are listed in the report generated by the BUILD utility.

• **error-count**  
The number of errors to detect before stopping.  
If you are doing a complete BUILD, you can specify different values for error-count for different kinds of errors.

• **CONTINUE**  
Indicates that processing should continue regardless of the number of errors detected.

• **STEPn**  
Directs the BUILD utility to perform only the nth step of the index or constraint building process.  
By default, if you do not specify a step, all four steps are performed as a single operation, and this is considered a complete BUILD.

• **STEP1**  
Directs the BUILD utility to perform only STEP1 of the BUILD process. STEP1 sweeps the area containing the specified table(s), creating an intermediate work file. The file contains the information needed later to build the index structures.  
If you specified the EXTRACT option in STEP1 of the LOAD utility, you do not need to run BUILD STEP1.  
In this case, the intermediate work file (SYS003) that is output from the LOAD utility can be used as the input file (SYS002) to STEP2 of the BUILD utility.

• **STEP2**  
Directs the BUILD utility to perform only STEP2 of the BUILD process. STEP2 determines the database key of the referenced table rows.

• **STEP3**  
Directs the BUILD utility to perform only STEP3 of the BUILD process. STEP3 creates the index structures needed for both indexes and constraints.

• **STEP4**  
Directs the BUILD utility to perform only STEP4 of the BUILD process. STEP4 updates the prefix(es) of the affected referencing table rows.
Usage

How to submit the BUILD statement

You submit the BUILD statement only through the batch command facility. You must run the batch command facility in local mode.

When to use BUILD

Use the BUILD utility after loading one or more tables using a phased or stepped LOAD.

You can also use the BUILD utility at any time to reorganize existing indexes on tables in an SQL-defined database.

When not to use BUILD

There is no need to run the BUILD utility if you loaded the table(s) with a complete LOAD. The indexes and constraints have already been built.

If the table is not part of an SQL-defined database, you cannot use the BUILD utility.

When to specify NO VALIDATE

The BUILD utility validates all referential constraints on the tables being worked on, not just the constraints currently being built. If all tables referenced by those specified by `table-id` have not yet been loaded, defer validation by specifying NO VALIDATE.

⚠️ Note: For more information and help in deciding which options to specify, see the Database Administration Section.

Sorting intermediate work files

If you run the load process in steps or phases, use the sort parameters in the SYSPCH file to sort the intermediate files.

JCL Considerations

When you submit a BUILD statement through the batch command facility, the JCL to execute the facility must include statements to define:

- The dictionary containing table definitions
- The files containing the tables and indexes to be processed
- Intermediate work files to be used by BUILD
• Sort work files, if doing a complete BUILD

**BUILD utility uses intermediate work files**

Each step of the build process, except BUILD STEP4, produces intermediate work files to be used by the next step. If you run a complete BUILD without separating steps, data is sorted in the intermediate files between the steps automatically. If you run a stepped BUILD, you must run the intermediate sorts.

⚠️ **Note:** When running a complete BUILD, SYS002 and SYS003 must point to the same intermediate file. If the database being processed is so large that the intermediate file must be a multi-volume file, it is required that all extents of the file be physically allocated prior to the initiation of the BUILD utility. If this cannot be done, then run a stepped BUILD. When running a stepped BUILD, SYS002 and SYS003 must point to different intermediate files. The data that is output in SYS003 by each step is input to the next step in SYS002.

The following table shows the output of the steps of the BUILD process:

<table>
<thead>
<tr>
<th>Step</th>
<th>Output</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>SYS003</td>
<td>For each record: (MAX SORT CONTROL KEY SIZE) + (MAX FOREIGN KEY SIZE) + 24</td>
</tr>
<tr>
<td></td>
<td>SYSPCH contains sort parameters</td>
<td>80 bytes</td>
</tr>
<tr>
<td>Step 2</td>
<td>SYS003</td>
<td>For each record: (MAX SORT CONTROL KEY SIZE) + (MAX FOREIGN KEY SIZE) + 24</td>
</tr>
<tr>
<td></td>
<td>SYSPCH contains sort parameters</td>
<td>80 bytes</td>
</tr>
<tr>
<td>Step 3</td>
<td>SYS003</td>
<td>For each prefix: 56 bytes</td>
</tr>
<tr>
<td></td>
<td>SYSPCH contains sort parameters</td>
<td>80 bytes</td>
</tr>
</tbody>
</table>

⚠️ **Note:** For more information about the generic JCL used to execute the batch command facility, see the section specific to your operating system.

**Examples**

The following example instructs the BUILD utility to perform a complete BUILD for the LOAD.M and LOAD.M2 sample tables. The NO VALIDATE option specifies that a VALIDATE should not be performed and ERRORS CONTINUE indicates that processing should continue regardless of the number of errors.
Sample Output

The following report was generated after executing the BUILD statement in the previous example.

```
IDMSBCF IDMS Batch Command Facility

*DEBUG IDMS OFF
CONNECT TO SYSCAT;
Status = 0
SET BATCH
   HEADINGS OFF WIDTH PAGE 79 UNDERLINE '
   SQLCODE ERROR
   COMPRESS ON;

UNLOCK AREA SYSSQL.DDLCAT;
Status = 1 Extended Reason Code = 2367 Messages follow:
DB002367 C1M353: Area SYSSQL.DDLCAT was not locked.
UNLOCK AREA SYSSQL.DDLCATX;
Status = 1 Extended Reason Code = 2367 Messages follow:
DB002367 C1M353: Area SYSSQL.DDLCATX was not locked.
-- **** Load data into Tables ****
*DEBUG IDMS ON

BUILD FOR LOAD.M,
   LOAD.M2
   ERRORS CONTINUE
   NO VALIDATE;
IDMSLOAD - volser SWEEP DATABASE yy-mm-dd-hh.mm.ss
IDMSLOAD - 3 records processed for table LOAD.M
IDMSLOAD - 3 intermediate records for index LOAD.IX_M
IDMSLOAD - 3 records processed for table LOAD.M2
IDMSLOAD - 3 intermediate records for index LOAD.IX1_M2
IDMSLOAD - 15 intermediate records were written to SYS003
IDMSLOAD - largest SYS003 record size is 56 characters
IDMSLOAD - SWEEP DATABASE processing completed
IDMSLOAD - volser CONNECT UP INDEXES yy-mm-dd-hh.mm.ss
IDMSLOAD - 6 intermediate records were written to SYS003
IDMSLOAD - largest SYS003 record size is 56 characters
IDMSLOAD - CONNECT UP INDEXES processing completed

AutoCommit will COMMIT transaction

Command Facility ended with warnings
```

CLEANUP

The CLEANUP utility physically erases logically deleted records from all or some areas in a database segment.

The CLEANUP utility works only with non-SQL-defined databases.

This article describes the following information:

- Authorization (see page 50)
Authorization

<table>
<thead>
<tr>
<th>To</th>
<th>You need this privilege</th>
<th>On</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clean up an area</td>
<td>DBAWRITE</td>
<td>The area</td>
</tr>
<tr>
<td>Clean up a segment</td>
<td>DBAWRITE</td>
<td>All areas within the segment</td>
</tr>
</tbody>
</table>

Syntax

```
CLEANUP SEGMENT segment-name
   USING subschema-name
   AREA area-name
```

Parameters

- **SEGMENT**
  Specifies the segment containing the areas to be processed.

- **segment-name**
  The name of the segment.

- **USING**
  Specifies a subschema that describes all the sets, records, and areas that are related to the logically deleted records being processed.

- **subschema-name**
  The name of the subschema.

- **AREA**
  Specifies one or more areas within the segment to process.
  By default, if you do not specify any areas, all areas in the specified segment are processed.

- **area-name**
  The name of an area within the specified segment.
Usage

How to submit the CLEANUP statement

You submit the CLEANUP statement by using either the batch command facility or the online command facility. The batch command facility can run in local mode or under central version.

How CLEANUP works

An area sweep is performed on each specified area. When a logically deleted record is found, each set in which the record is a member is processed. Every logically deleted record in the set is first disconnected and then physically erased. When all sets in which the original record was a member are processed, the sweep of the area is resumed.

When to use CLEANUP

Use the CLEANUP utility:

- To erase deleted records in sets with no prior pointers
- Before running the RESTRUCTURE utility
- Before running the UNLOAD or RELOAD utilities

When not to use CLEANUP

Records in sets with prior pointers are erased when they are deleted. If all the sets in an area have prior pointers, you never need to use the CLEANUP utility.

Journaling

You can use journaling while executing the CLEANUP utility to allow for recovery with the ROLLBACK utility.

LDEL counts

The number of logically deleted records (LDELS) reported in messages UT011020 and UT011021 reflect the number of LDELS encountered by the CLEANUP utility. Due to the architecture of how LDELS are removed, multiple LDELS might get removed for each one counted by the utility. To obtain an accurate count of the number of LDELS processed by the CLEANUP utility, a DBAN or PRINT SPACE report should be created for the requested area(s), both before and after the execution of the CLEANUP utility.

JCL Considerations

When you submit a CLEANUP statement through the batch command facility, the JCL to execute the facility must include statements to define:

- The files containing the areas to be processed.
The journal files of the DMCL you are using. If you are not journaling, these should be dummied out.

⚠️ **Note:** For more information about the generic JCL used to execute the batch command facility, see the section for your operating system in this section.

Example

The following example directs the CLEANUP utility to erase any logically deleted records in the EMPDEMO-REGION area.

```
cleanup segment empdemo using empss01
    area emp-demo-region;
```

Sample Output

When the CLEANUP operation is completed, the following listing is provided.

```
IDMSBCF mm/dd/yy PAGE 1
IDMS Batch Command Facility

CLEANUP SEGMENT EMPDEMO USING EMPSS01
    AREA EMPDEMO.EMP-DEMO-REGION;
UT000038 Starting CLEANUP of area EMPDEMO.EMP-DEMO-REGION
UT011020 CLEANUP completed. Pages read=10  Records read=5  Ldel records read=2
UT011021 LDEL Record name BB  Found=2  Removed=2
Status = 0

AutoCommit will COMMIT transaction
```

⚠️ **Note:** For more information about preventing logically deleted records, see the *CA IDMS Database Administration Section*.

**CONVERT CATALOG**

The CONVERT CATALOG utility converts the SYSTEM schema within a catalog to reflect the definition specified by the SYSTEM schema DDL delivered with the current release of CA IDMS.

**Authorization**

None
Syntax

►►─── CONVERT CATALOG ──────────────────────────────────────►◄

Usage

When to convert a catalog

After installing a new version of CA IDMS, you should convert every catalog that contains a definition of the SYSTEM schema. Typically, deferring the conversion of a catalog will result in the inability to retrieve new information using SQL queries against the catalog. It may also prevent certain CA IDMS Visual DBA functions from executing correctly.

How to convert a catalog

Through either the online or batch command facility, connect to the dictionary containing the catalog to be converted and issue the CONVERT CATALOG command. If executing the batch command facility in local mode, be sure to backup the catalog before converting it.

The CONVERT CATALOG utility can be executed more than once against the same catalog. if the target catalog has previously been converted to the current release format, no updates take place.

Prior release considerations

The CONVERT CATALOG utility can be used to upgrade a catalog from any prior release of CA IDMS starting with 12.0. Should fallback to the earlier release be necessary after a catalog is converted, there is usually no special action needed with regard to the catalog. However, check each intervening CA IDMS Release Summary for any special considerations that might apply when upgrading from a specific release.

Reporting successful execution

After successful execution, CA IDMS issues one of the following informational messages to indicate the status of the conversion:

- If a catalog conversion is performed, the message indicates the number of rows of each type that are changed.
- If a catalog conversion is not required, an appropriate message is issued.

CONVERT PAGE

The CONVERT PAGE utility changes the page range for an area or changes the maximum number of records that can be stored on a page of an area.

This article describes the following information:

- Authorization (see page 54)
Authorization

<table>
<thead>
<tr>
<th>To</th>
<th>You Need This Privilege</th>
<th>On</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change the page range for an area or the maximum number of records that can be stored on a page of an area</td>
<td>DBAWRITE</td>
<td>All areas being converted</td>
</tr>
</tbody>
</table>

Syntax

```
CONVERT PAGE IN SEGMENT source-segment TO target-segment

DMCL target-dmcl

FILE source-segment. INTO new-ddname

NOTIFY INTERVAL not-interval

AREA source-seg.area TO target-seg .area
```

Parameters

- **source-segment**
  Names a segment to be processed. All areas in this segment are compared to matching areas in the target segment for page range and maximum records per page changes. Unless restricted by the FILE parameter, all files in this segment are scanned for page range changes and copied to new files.

- **source-seg.area**
  Names an area to be processed. All specified areas are compared to the corresponding target areas and checked for page range and maximum records per page changes. Unless restricted by the FILE parameter, all files for this area are scanned for page range changes and copied to new files.
- **target-segment**
  Names the segment with the new page range and maximum records per page definitions. If you omit target-segment, it defaults to the source segment name. You should use this when the target segment is in the same DMCL as the source.

- **target-seg.area**
  Names the area that contains the new page range and maximum records per page definitions. If you omit this, it defaults to source-seg.area name. If you omit the area, it defaults to the source area name. You should use this when the target area is in the same DMCL as the source.

- **target-dmcl**
  Names the DMCL that contains the target segment or area definitions. Use this parameter when the target segment or area is defined in a different DMCL than the current one.

- **filename**
  Restricts the conversion to the selected source files. If you do not specify the FILE parameter, then all files for selected segments and areas are converted. However, all target file definitions must have ddnames that are different from the source file definitions.

- **new-ddname**
  Names the JCL ddname used for the output file. This name must be unique in the job step. If you don’t specify new-ddname, the ddname in the target file definition is used. In this case the name of the target file must match the source file.

- **not-interval**
  Specifies the notify interval. After every not-interval record read, a message is issued to the console stating how far the job has progressed. If omitted, the default interval is 10,000 records. If specified as zero (0), no notify message is issued.

**Usage**

You can process an entire segment or one or more individual areas using this utility. The utility first identifies the changes that need to be made by comparing old and new definitions for the segment or areas. Once all changes have been identified, it converts one or more files. It is possible to run the utility in parallel to process several files concurrently.

This utility allows you to reassign the page range of an area so that you can consolidate free pages. It also allows you to change the maximum number of records on a page permitting more effective use of the space on a page and more flexibility in choosing page sizes.

To use the feature, you must define one or more new segments and include them in a DMCL, or alter the definition of an existing segment and include it in a new DMCL. The utility requires both the old and new definitions of the affected segment or areas to be available at runtime.

Once the definitions are available, you can execute the utility. There must be enough disk space available to hold all converted files. The converted files can also be written to tape and then copied back to the original database files. Be sure to back up the original database beforehand.

Once all files have been converted, you must make the new segment or area definitions effective.

**Page range and radix changes**
Page range and radix changes are determined by comparing all source and target area definitions. Regardless of which files are selected for conversion, all differences are applied to all files selected for conversion.

**File conversion**

File conversion is performed against all files in the specified segment or areas, unless the FILE parameter is specified, in which case only specified files are converted.

File conversion consists of copying a page from the source file to the target file while making any required changes. Page number changes are accomplished by subtracting the old low page and adding the new low page. Maximum records per page changes are effected by breaking up a dbkey into a line and page number using the old information and reassembling it using the new information.

**Note:** If no changes apply to a selected file it is copied anyway.

The CONVERT PAGE utility also has the following usage considerations:

- If an area that is being converted has a cross-area set or linked constraint, then you must convert all areas touched by the set or link. You must determine what files are affected, and therefore what files must be converted.

- If you need to convert multiple files, the files do not all have to be converted in the same job.

- As long as all changes are identified to each job, you can convert each affected file separately. It may be desirable to run several jobs in parallel to reduce the time required to convert an area.

**Important!** It is important that you identify all page range changes that affect a file before the file is converted. Depending on the nature of the page range change, it may not be possible to apply a partial change to a previously converted file.

- If you omit a change identification, then you will need to reapply all changes to the original file, not the converted file.

- You cannot change the number of pages assigned to an area, nor lower the maximum number of records per page to less than the actual number of records stored on any page of an area.

- This utility must run in local mode and with update activity quiesced on the affected areas.

- The source and target files must match in page size and number of blocks. (The source file is copied block for block.) If the target file does not match then the converted file will not be usable by the new file definition.
JCL Considerations

All input and output ddnames must be unique within the same job step. If the target DMCL files do not have unique ddnames, use the INTO option for each output file.

JCL must include DD statements for the output files. If a file is not defined to use dynamic file allocation, then DD statements for the input files are also required.

**Note:** For more information about the generic JCL used to execute the batch command facility, see the section for your operating system in this section.

Examples

The EMPDEMO database has three areas: the Emp-demo-region, Org-demo-region, and the Ins-demo-region. There are cross-area sets between the Emp and Org-demo-regions, and between the Emp and Ins-demo-regions, but there are no sets between the Ins-demo-region and the Org-demo-region; therefore, if you need to change the:

- Page range for the Org-demo-region, you must convert both the Emp-demo-region and the Org-demo-region.
- Page range for the Ins-demo-region, you must convert both the Emp-demo-region and the Ins-demo-region.
- Page range for the Emp-demo-region, you need to convert all three areas.

Example 1

The first step in converting an area's page range is to define the new segment with all changes. This new segment could be included in the current DMCL if its name is different from the old segment, if the page group and page ranges are different from the old ones, and if the ddnames are different from the old ones.

After the DMCL is updated, the required syntax to do all conversions would be:

```
CONVERT PAGE IN SEGMENT EMPDEMO TO NEWDEMO;
```

The CONVERT PAGE utility scans both segments and determines which areas are being changed. Because the output files have unique ddnames, they do not need to be identified, but all three files that are in the EMPDEMO segment are converted, even if the changes do not affect all files.

To restrict the process, you could code the following:

```
CONVERT PAGE IN SEGMENT EMPDEMO TO NEWDEMO
   FILE EMPDEMO-FILE, ORGDEMO-FILE;
```
This syntax would only convert the EMPDEMO and ORGDEMO files. If the INSDEMO file were not affected by any changes, then this would be all that would be needed. However, if the INSDEMO file were affected, it could be converted with a separate job as follows:

```plaintext
CONVERT PAGE IN SEGMENT EMPDEMO TO NEWDEMO
   FILE INSDEMO-FILE;
```

This works because all changes were identified to both jobs, because all changes are contained in the single segment.

When completed, the DMCL would have to be modified a second time to remove the old segment definition and to possibly rename the new segment to the old one, to possibly change the new page group to the old page group, and to possibly change the ddnames back to the old ones.

**Example 2**

Another approach is to define a new DMCL with all changes made to the existing segment.

In this case you would not have to modify the DMCL a second time, but more syntax is required for the conversion process, as follows:

```plaintext
CONVERT PAGE IN SEGMENT EMPDEMO TO EMPDEMO DMCL NEWDMCL
   FILE EMPDEMO-FILE INTO EMPDDX,
      INSDEMO-FILE INTO INSDDX,
      ORGDEMO-FILE INTO EMPDDX;
```

This syntax requires that you give each output file a unique ddname for the run only, because the defined ddnames in the target DMCL are the same as the source DMCL. However, once converted, the old DMCL is discarded, and the new one is renamed and the old one is replaced.

Again, if all three files are not affected, the number of files converted can be reduced by not naming the unaffected files.

**Example 3**

If area syntax were being used, the syntax could be as follows:

```plaintext
CONVERT PAGE IN AREA
   EMPDEMO.EMP-DEMO-REGION TO NEWDEMO,
   EMPDEMO.INS-DEMO-REGION TO NEWDEMO,
   EMPDEMO.ORG-DEMO-REGION TO NEWDEMO;
```

Or if the areas were in a different DMCL:

```plaintext
CONVERT PAGE IN AREA EMPDEMO.EMP-DEMO-REGION,
   EMPDEMO.INS-DEMO-REGION,
   EMPDEMO.ORG-DEMO-REGION
   DMCL NEWDMCL
   FILE EMPDEMO-FILE INTO EMPDDX,
      INSDEMO-FILE INTO INSDDX,
      ORGDEMO-FILE INTO EMPDDX;
```

This syntax works because all areas affected by the changes are identified. Again, this converts all files even if some are not affected.

**Example 4**

To only convert affected files, specify only the files that should be converted:
CONVERT PAGE IN AREA EMPDEMO.EMP-DEMO-REGION TO NEWDEMO,
EMPDEMO.INS-DEMO-REGION TO NEWDEMO,
EMPDEMO.ORG-DEMO-REGION TO NEWDEMO
FILE EMPDEMO-FILE, INSDEMO-FILE;

Or if the areas were in a different DMCL:
CONVERT PAGE IN AREA EMPDEMO.EMP-DEMO-REGION,
EMPDEMO.INS-DEMO-REGION,
EMPDEMO.ORG-DEMO-REGION
DMCL NEWDMCL
FILE EMPDEMO-FILE INTO EMPDDX,
INSDEMO-FILE INTO INSDDX;

This syntax would only cause the Emp-demo-region and the Ins-demo-region to be converted.

If you decided that the Org-demo-region also needed to be converted, you could run the following:

CONVERT PAGE IN AREA EMPDEMO.EMP-DEMO-REGION,
EMPDEMO.INS-DEMO-REGION,
EMPDEMO.ORG-DEMO-REGION
DMCL NEWDMCL
FILE ORGDEMO-FILE INTO ORGDDX;

This syntax works because all three areas were identified to compare phase. Even though the files were converted in separate jobs, all jobs knew about all changes.

Bad Example 4

The wrong way to do the previous example would be to reduce the areas considered for change, instead of the files to be converted:

CONVERT PAGE IN AREA EMPDEMO.EMP-DEMO-REGION,
EMPDEMO.INS-DEMO-REGION,
DMCL NEWDMCL
FILE EMPDEMO-FILE INTO EMPDDX,
INSDEMO-FILE INTO INSDDX;

This syntax works only if the Org-demo-region is unaffected by any changes. Only the Emp-demo-region and the Ins-demo-region are converted. But if it were later discovered that the Org-demo-region required conversion, you would have to reconvert the Emp-demo-region.

CONVERT PAGE IN AREA EMPDEMO.EMP-DEMO-REGION,
EMPDEMO.INS-DEMO-REGION,
EMPDEMO.ORG-DEMO-REGION
DMCL NEWDMCL
FILE EMPDEMO-FILE INTO EMPDDX,
ORGDEMO-FILE INTO ORGDDX;

\[\textbf{Note:}\quad \text{Ins-demo-region did not have to be reconverted because it was unaffected by changes to the Org-demo-region, but because the Emp-demo-region is affected by changes to the Ins-demo-region; Ins-demo-region is included in the list of changed areas.}\]
Sample Output

IDMSBCF nn.nn CA IDMS Batch Command Facility mm/dd/yy PAGE 1

CONVERT PAGE
  AREA EMPDEMO.EMP-DEMO-REGION TO EMPXDEMO,
  EMPDEMO.ORG-DEMO-REGION TO ORGXDEMO,
  EMPDEMO.INS-DEMO-REGION TO INSXDEMO
  DMCL EMPXDMCL
  NOTIFY 10

UT018000 Convert Page Utility Starting:
UT018001 Source DMCL: LRDTDMCL Target DMCL: EMPXDMCL
UT018003 Comparing source area: EMPDEMO.EMP-DEMO-REGION
  with target area: EMPXDEMO.EMP-DEMO-REGION
UT018012 Area added to change table.
UT018013 SOURCE AREA Range: 75,001 to 75,100 Radix: 8
UT018013 TARGET AREA Range: 75,001 to 75,100 Radix: 7
UT018003 Comparing source area: EMPDEMO.ORG-DEMO-REGION
  with target area: ORGXDEMO.ORG-DEMO-REGION
UT018012 Area added to change table.
UT018013 SOURCE AREA Range: 75,151 to 75,200 Radix: 8
UT018013 TARGET AREA Range: 75,001 to 75,050 Radix: 9
UT018003 Comparing source area: EMPDEMO.INS-DEMO-REGION
  with target area: INSXDEMO.INS-DEMO-REGION
UT018012 Area added to change table.
UT018013 SOURCE AREA Range: 75,101 to 75,150 Radix: 8
UT018013 TARGET AREA Range: 75,001 to 75,050 Radix: 8
UT018005 File EMPDEMO.EMPDEMO has been selected for processing
UT018006 Source DName: EMPDEMO Target DName: EMPXDEMO
UT018005 File EMPDEMO.INSDEMO has been selected for processing
UT018006 Source DName: INSDEMO Target DName: INSXDEMO
UT018005 File EMPDEMO.ORGDEMO has been selected for processing
UT018006 Source DName: ORGDEMO Target DName: ORGXDEMO
Starting CONVERSION of file EMPDEMO.EMPDEMO
UT001006 Program IDMSUCON processed 10 records
UT001006 Program IDMSUCON processed 20 records
UT001006 Program IDMSUCON processed 30 records
UT001006 Program IDMSUCON processed 40 records
UT001006 Program IDMSUCON processed 50 records
UT001006 Program IDMSUCON processed 60 records
UT001006 Program IDMSUCON processed 70 records
UT001006 Program IDMSUCON processed 80 records
UT001006 Program IDMSUCON processed 90 records
UT001006 Program IDMSUCON processed 100 records
CONVERSION complete
  Pages Read: 100 Modified: 100
  Records Read: 346 Modified: 346
  Dbkeys Read: 2,847 Modified: 2,847 Null: 1
  Errors Page: 0 Dbkey: 0
Starting CONVERSION of file EMPDEMO.INSDEMO
UT001006 Program IDMSUCON processed 110 records
UT001006 Program IDMSUCON processed 120 records
UT001006 Program IDMSUCON processed 130 records
UT001006 Program IDMSUCON processed 140 records
UT001006 Program IDMSUCON processed 150 records
CONVERSION complete
  Pages Read: 50 Modified: 50
  Records Read: 84 Modified: 84
  Dbkeys Read: 494 Modified: 494 Null: 0
  Errors Page: 0 Dbkey: 0
Starting CONVERSION of file EMPDEMO.ORGDEMO
UT001006 Program IDMSUCON processed 160 records
UT001006 Program IDMSUCON processed 170 records
UT001006 Program IDMSUCON processed 180 records
UT001006 Program IDMSUCON processed 190 records
UT001006 Program IDMSUCON processed 200 records
CREATE DSMODEL

The CREATE DSMODEL utility defines a temporary model of data set attributes to use for dynamic file allocation in conjunction with the REORG utility.

This article describes the following information:

- Authorization (see page 61)
- Syntax (see page 61)
- Parameters (see page 62)
- Usage (see page 63)
- Example (see page 64)

Authorization

None.

Syntax

```
CREATE DSMODEL - model-name FROM existing-model-name dataset-attribute-spec
```

Expansion of `dataset-attribute-spec`

```
DSN - dsn-expression
UNIT - unit-name
VOLSER - volser
VOLCNT - volume-count
SPACE - CYL (primary T )
```

More Information

- For more information about page ranges, see the CA IDMS Database Administration Section.
- For more information about max records per page, see the CA IDMS Database Administration Section.
Expansion of *dsn-expression*

```
<table>
<thead>
<tr>
<th>character-string</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsn-variable</td>
</tr>
</tbody>
</table>
```

Expansion of *dsn-term*

```
<table>
<thead>
<tr>
<th>character-string</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsn-variable</td>
</tr>
</tbody>
</table>
```

Expansion of *dsn-variable*

```
<table>
<thead>
<tr>
<th>dsn-variable-name</th>
</tr>
</thead>
<tbody>
<tr>
<td>( start , length )</td>
</tr>
</tbody>
</table>
```

Expansion of *dsn-variable-name*

```
&DD.                `
- **CYL/TRK/blksize**
  Specifies the allocation unit for a disk data set. CYL allocates the file in units of cylinders. TRK allocates the file in units of tracks. Blksize allocates the file in units whose size is blksize.

- **primary**
  Specifies the number of units in the primary allocation for a disk file.

- **secondary**
  Specifies the number of units in the secondary allocation for a disk file.

- **block-size**
  Specifies the blocksize to be used in creating (and accessing) a new file.

- **number-of-buffers**
  Specifies the number of buffers to be used in accessing a file.

- **data-class, management-class, storage-class, dataset-type**
  Specifies a data set’s SMS data class, management class, storage class, and dataset type respectively.

- **dsn-variable-name**
  Specifies the name of a symbolic variable whose value can be used in whole or in part as a substitution parameter in the construction of a data set name.

  - **&DD.**
    Specifies the DDNAME (linkname) through which the data set is referenced.

- **start**
  Specifies the position of the first character in the value of the symbolic variable that is to be used in the construction of the data set name. Start must not identify a position beyond the last non-blank character in the value.

- **length**
  Specifies the number of characters in the value of the symbolic variable that are to be used in the construction of the data set name. If not specified, length defaults to the number of non-blank characters from the start character to the end of the value. Trailing blank characters are truncated.

### Usage

#### Using data set models

Data set models are currently used only to specify attributes of the work files used by the REORG utility. For more information about how it uses data set models, see [REORG](#) (see page 200).

#### Data set model duration

Data set models are transient entities that, once created, exist in memory for the life of the database session in which they are created. They are not stored in a dictionary.
Defining dummy files

You can define attributes for a dummy file by creating a data set model whose dsn-expression resolves to the character string NULLFILE. This is equivalent to specifying DUMMY on a DD JCL statement.

Example

The following example defines models for work files that are used by the REORG utility.

CREATE DSMODEL W* DSN 'DBDC.TEMP.ZIP9.&DD'. UNIT SYSDA SPACE CYL (1000,100) BLKSIZE 8192;
CREATE DSMODEL WD* FROM W* SPACE CYL (10, 10) VOLSER DBA001 UNIT 3390;
CREATE DSMODEL WS* FROM W* SPACE CYL (3000, 10);
REORG ...

By default, all work files are allocated on SYSDA devices with a 1000 cylinder initial space allocation, have a block size of 8192, and a data set name of "DBDC.TEMP.ZIP9.xxxxxxx", where xxxxxxxx is the file's DDNAME. DBKEY files (those whose DDNAME begins with "WD") are allocated on volume DBA001 with an initial allocation of 10 cylinders. Sorted files (those whose DDNAME begins with "WS") have an initial allocation of 3000 cylinders.

EXPAND PAGE

The EXPAND PAGE utility increases the page size of an area by transferring a database file to a new file with an expanded block size.

This article describes the following information:

- Authorization (see page 64)
- Syntax (see page 65)
- Parameters (see page 65)
- Usage (see page 65)
- JCL Considerations (see page 66)
- Example (see page 66)
- Sample Output (see page 67)

Authorization

<table>
<thead>
<tr>
<th>To</th>
<th>You need this privilege</th>
<th>On</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expand the pages of a file</td>
<td>DBAWRITE</td>
<td>All areas that map to the file.</td>
</tr>
</tbody>
</table>
Syntax

```
EXPAND page for FILE segment-name.file-name
INTO ddname
NEWSIZE new-page-size
```

Parameters

- **FILE**
  Specifies the file whose pages are to be expanded.

- **segment-name**
  The name of the segment associated with the file.

- **file-name**
  The name of the file.

- **INTO**
  Specifies the external name of the output file.

- **ddname**
  The external z/OS or CMS name of the new file.

- **filename**
  The external z/VSE name of the new file.

- **NEWSIZE**
  Specifies the new page size.

- **new-page-size**
  The new page size, in bytes. The new page size must be a multiple of four and greater than the current page size.

Usage

How EXPAND PAGE works

A page at a time from the old file is read. The page is expanded adding the new space before the footer section of the page. The space management entry for the page is adjusted to reflect the added space. The space management pages are also expanded, but they continue to have the same number of entries as before.

How to submit the EXPAND PAGE statement
You submit the EXPAND PAGE statement only through the batch command facility. You must be running CA IDMS/DB in local mode. You must also vary affected areas to retrieval mode or offline in all currently executing DC/UCF systems.

**If an area maps to more than one file**

You can expand pages in only one file at a time. If you are expanding the pages for an area that maps to more than one file, you must run the EXPAND PAGE utility once for each file the area is associated with.

**Regenerate DMCL modules**

After running the EXPAND PAGE utility, you must:

- Alter the page size of the area to the new page size in all files where the area is used. Please remember that you should only change the PAGE SIZE on the ALTER AREA segment, because altering the PRIMARY SPACE even to the current value can result in any EXTEND SPACE attribute being removed.

- Regenerate all DMCLs containing the file’s segment.

The pages are expanded but the space management pages are not relocated. Therefore, when you redefine an area whose page size has been expanded, you must use the ORIGINAL PAGE SIZE clause of the AREA statement.

**JCL Considerations**

When you submit an EXPAND PAGE statement through the batch command facility, the JCL to execute the facility must include statements to define the files to be processed.

⚠️ **Note:** For more information about the generic JCL used to execute the batch command facility, see the section for your operating system in this section.

**Example**

The following example directs the EXPAND PAGE utility to increase the size of the EMPDEMO. EMPDEMO file.

```
expand file empdemo.empdemo
into newfile
newsize 4820;
```
Sample Output

The EXPAND PAGE utility produces the following listing after successful completion of the statement in the previous example.

```
IDMSBCF mm/dd/yy PAGE 1
EXPAND FILE EMPDEMO.EMPDEMO INTO NEWFILE NEWSIZE 4820;
OUT000039 Starting Expansion of file EMPDEMO.EMPDEMO
OUT015007 SMI based on 4,276 characters for area EMPDEMO.EMP-DEMO-REGION
OUT015008 Low page 75,001 High page 75,100 Page group 0
OUT000040 Expansion complete
Status = 0
AutoCommit will COMMIT transaction
Command Facility ended with no errors or warnings
```

Note: For more information about recompiling DMCL modules and for sectionlines on when to use the EXPAND PAGE utility, see the CA IDMS Database Administration Section.

EXTRACT JOURNAL

The EXTRACT JOURNAL utility conceptually extracts the most recent AFTR image for each dbkey recorded on an archived journal file and writes it to an extract file. The extract file can later be used as input to a ROLLFORWARD command for a "quick" recovery of a database area or file.

When used as part of a regular procedure to extract journal images from archived journal tapes, it allows for a quicker recovery during a critical time frame by doing much of the work during a non-critical time frame. It also speeds up recovery by extracting only the journal images it needs to recover the areas or files selected.

This article describes the following information:

- Authorization (see page 67)
- Syntax (see page 68)
- Parameters (see page 68)
- Usage (see page 71)
- JCL Considerations (see page 74)
- Example (see page 74)
- Sample Output (see page 74)

Authorization

DBAWRITE authority is needed for all options.
Syntax

EXTRACT JOURNAL FOR DMCL
  FILE segment-name.file-name
  AREA segment-name.area-name
  SEGMENT segment-name

COMPLETED ALL TO ddname
  file-name
  link-name
  SYS002

VERIFY QUIESCE
  NONE
  LIMITED
  FULL

UTC

START AT ' date-time '
  date
  time

STOP AT ' date-time '
  date
  time

Parameters

- **DMCL**
  Specifies that dbkeys for all areas defined in the DMCL specified in the SYSIDMS parameter file will be included in the extract file.

- **FILE**
  Specifies to include `segment-name.file-name` dbkeys in the specified file in the extract file.

- **AREA**
  Specifies to include `segment-name.area-name` dbkeys in the specified area in the extract file.

- **SEGMENT**
  Specifies to include dbkeys for all areas defined in the specified segments in the extract file.

- **COMPLETED**
  If you do not specify a STOP time, only after images for transactions that have completed by the end of the archive file will be written to the extract file. After images are written to the extract file for incomplete distributed transactions whose state is InDoubt at the end of the journal file (unless a section recovery control file entry overrides this behavior by specifying BACKOUT for one or more of the transactions).
  If you do specify a STOP time, only after images for transactions that have completed by the first
checkpoint at or following the stop time will be written to the extract file. After images are written to the extract file for incomplete distributed transactions whose state is InDoubt at the stop time (unless a section recovery control file entry overrides this behavior by specifying BACKOUT for one or more of the transactions).

- **ALL**
  If you do not specify a STOP time, after images for all transactions will be written to the extract file.
  If you do specify a STOP time, after images for all transactions will be written to the extract file up to the point at or after the stop time where there are no active transactions. If such a point is not found by the end of the archive file, all after images will be written.

- **TO**
  Specifies the external name to be used for the extract file.

- **ddname**
  The external z/OS, or CMS name for the extract file.

- **file-name**
  The external z/VSE name for the extract file.

- **SYS002**
  The default name for all systems.

- **VERIFY QUIESCE**
  Specifies the level of quiesce point verification that will be performed.

  - **NONE**
    Specifies that no quiesce point verification should be done. This is the default if neither VERIFY QUIESCE nor START AT is specified. If START AT is specified, NONE is treated as LIMITED.

  - **LIMITED**
    Specifies that limited quiesce point verification be performed. This is the default if VERIFY QUIESCE is not specified and START AT is specified.

  - **FULL**
    Specifies that the strictest form of quiesce point verification be performed. This is the default if VERIFY QUIESCE is specified without indicating the verification level.

- **START AT**
  Specifies that the operation should start only after reaching a specified date and time in the journal file. Only images for transactions that begin after the specified start time will be processed. By default, processing starts at the beginning of the journal file.

- **STOP AT**
  Specifies that the operation should stop as soon as possible after reaching a specified date and time in the journal file.
  By default, processing stops when the end of the journal file is reached.

- **date**
  Specifies the date, in one of the following formats:
  - yyyy-mm-dd
In these formats, the following rules apply:

- **yyyy** specifies the year. *yyyy* must be an integer in the range 0001 through 9999. Leading zeros are optional.

- **mm** specifies the month within the year. *mm* must be an integer in the range 01 through 12. Leading zeros are optional.

- **dd** specifies the day within the month. *dd* must be an integer in the range 01 through 31. Leading zeros are optional.

The combined values of *yyyy*, *mm*, and *dd* must represent a valid date. For example, 1988-02-29 is a valid date. 1989-02-29 is not.

**date-time**

Specifies the date and time, where:

- The format for specifying the DATE-TIME is:
  
  *yyyy-mm-dd-hh.mm.ss.ffffff*

- The rules for specifying the DATE component of DATE-TIME are the same as for DATE. The rules for specifying the TIME component of DATE-TIME are:

  - **hh** specifies the hour on a 24-hour clock. *hh* must be an integer in the range 00 through 23. Leading zeros are optional.

  - **mm** specifies the number of minutes past the hour. *mm* must be an integer in the range 00 through 59. Leading zeros are optional.

  - **ss** specifies the number of seconds past the minute. *ss* must be an integer in the range 00 through 59. Leading zeros are optional.

  - **ffffff** specifies the number of millionths of a second past the specified second. *ffffff* is optional; if you include it, it must be an integer in the range 000000 through 999999. The default value is 000000. Trailing zeros are optional.

**time**

Specifies the time in one of the following formats:

- **hh.mm.ss**

- **hh:mm:ss**

The rules for specifying TIME are the same as those listed for DATE-TIME. When TIME is specified, the date defaults to the current date.

**UTC**

Specifies that Start and Stop times are interpreted as UTC times instead of local times.
Usage

How to submit EXTRACT JOURNAL

EXTRACT JOURNAL must be submitted through the batch command facility and in local mode.

When to use EXTRACT JOURNAL

EXTRACT JOURNAL is most commonly used on a daily basis or after each archive journal is created. It is used as part of a plan for recovering a file or area on a device that takes a hardware hit.

⚠️ Note: For more information, see the ‘FROM EXTRACT’ option on the ROLLFORWARD command. It allows ROLLFORWARD processing to be done during a recovery situation when time is critical or extract processing to be done during a non-critical time in case it is needed.

How EXTRACT JOURNAL works

EXTRACT JOURNAL works much the same as ROLLFORWARD using the sort option, except that instead of applying the latest after image directly to the database, the image is written to the extract file. The input archive journal is read and after images for selected areas are sorted. Only the latest image from a completed transaction is saved, and the rest are discarded. By saving only the latest after images for only selected areas, the resulting extract file is smaller and presorted, making subsequent ROLLFORWARD processing go much faster.

If a transaction does not end on the current set of input files, all AFTR images from that transaction are written to the extract file. When these extra images are processed by a subsequent EXTRACT JOURNAL or by a ROLLFORWARD command, they are included with completed AFTR images or discarded depending on the final status of that transaction. Only the most recent AFTR image taken from a completed transaction for a given dbkey is applied to the database.

Multiple input archive journals

Multiple input archive journals can be used as input provided they are read in the correct order. They do not need to be merged into a single file. With restrictions, each archived journal can also be processed with separate EXTRACT JOURNAL jobs, and the resulting extract files can be concatenated as input to one ROLLFORWARD job.

COMPLETE and ALL with STOP time considerations

If the COMPLETE option or the ALL option with a STOP time is specified, the resulting extract file will not contain any information for run units that were active at the end of the archive file. Therefore, an extract file created with these options should never be used as input with extract files created from subsequent archive journals. Use these options only if you intend this extract file to be the last in a series of extract files that will be used as input to a ROLLFORWARD.

Multiple output extract files from one archive journal
To create multiple extract files from one archive journal (one for each database segment, for instance) EXTRACT JOURNAL must be run multiple times, once for each group of database areas or files that you may wish to recover separately.

Since ROLLFORWARD has the ability to select images from an input extract file by area or file, you do not need an extract file for each area or file. For example, you can create an extract file for each critical database segment, and if a recovery is needed for one file in that segment, you can use ROLLFORWARD to recover just that file from the segment extract file. This speeds recovery while minimizing the number of extract tapes you need to maintain.

**KSDS native VSAM records**

EXTRACT JOURNAL does not support KSDS native VSAM records.

**Controlling the starting point of the extract operation**

If no START AT parameter is specified, the extract operation starts with the first journal image on the input archive file(s). If START AT is specified, the extract operation starts with the first checkpoint record (BGIN, COMT, ENDJ, ABRT, or CKPT) whose timestamp is on or after the specified time.

Specifying a start time may save recovery time and also circumvent issues associated with aborted recovery units that span the start of the input file. It further enables the extract operation to begin processing at a quiesce point that does not correspond with a journal file boundary.

If START AT is specified, quiesce point verification is always performed even if VERIFY QUIESCE NONE is specified. Limited verification will be performed unless full verification is specifically requested.

**Quiesce point verification**

A quiesce point is a point in time during which there is no update activity for some portion of a database. To perform a correct recovery, you must begin the recovery operation at a quiesce point for the portion of the database being recovered. To assist in this effort, the extract journal utility provides three levels of quiesce point verification: none, limited, and full.

None means that no quiesce point verification is performed. It is appropriate for extract operations whose input is not expected to coincide with a quiesce point. For example, if extracting journal information incrementally, then quiesce point verification could be used when processing the archive files produced immediately after a quiesce operation, but must not be used when processing subsequent archive files.

The limited and full options both enable quiesce point verification. They do this by checking for the existence of spanned recovery units that are recovery units that are active at the start of the extract operation. A recovery unit is represented by journal images starting with a BGIN or COMT checkpoint and ending with a COMT, ENDJ, or ABRT checkpoint. If a spanned recovery unit updates the specified portion of the database, then the extract operation is not starting at a quiesce point. If this situation is detected and either limited or full quiesce point verification is in effect, the extract operation will terminate with an error.
However, it is not always possible to know whether a spanned recovery unit affects the specified portion of the database or not. If the initial BGIN or COMT checkpoint record for a recovery unit is not contained on the archive file being processed, then it is not possible to determine whether it updated the specified portion of the database. Such a recovery unit is referred to as an indoubt recovery unit.

The time line illustrates what is meant by an indoubt recovery unit. The journal images for recovery unit $R$ are written to the journal file at the times shown. If the archive file includes images starting at $T_1$, then $R$ is not an indoubt recovery unit because the archive file contains all journal images written for $R$ since its inception. Similarly, if the archive file starts at time $T_3$, $R$ is not an indoubt recovery unit, because the archive file contains no images for $R$ whatsoever. However, if the archive file starts at time $T_2$, then $R$ is an indoubt recovery unit, since the archive file does not contain all journal images written since its inception.

If an indoubt recovery unit does not span the start of the rollforward operation, its existence doesn’t matter. But if an indoubt recovery unit is also a spanned recovery unit, then the extract operation might not be starting at a quiesce point.

The action taken if an indoubt spanned recovery unit is encountered depends on whether limited or full quiesce point verification is in effect. Under full verification, the extract operation will terminate with an error. Under limited verification, a warning message will be issued identifying the recovery unit, but processing will continue. Warning messages produced under limited verification should be examined to ensure that the identified recovery units in fact did not affect the specified portion of the database. If there is any doubt, the PRINT JOURNAL utility statement should be used to gain more information about the indoubt recovery units. If after researching the situation, it is found that an indoubt recovery unit did update the specified portion of the database, the resulting extract file must not be used for recovery purposes. You must locate a quiesce point corresponding to a backup of the specified portion of the database and begin the extract operation from that point.

**When to use full or limited verification**

Full quiesce point verification should only be used if you expect that no indoubt spanned recovery units are active at the starting point of the extract operation. The only way to guarantee this is to process the archive files that were created immediately following a quiesce of update activity across all areas. One way to establish such a quiesce point is to shut down a central version. Another way is to use the DCMT QUIESCE command and specify a DBNAME that includes every area in the DMCL.

Limited quiesce point verification can be used when processing the archive files produced immediately following a quiesce operation for the portion of the database for which the extract is being performed. One way to do this is to use the DCMT QUIESCE SEGMENT command to quiesce a segment and then use the limited quiesce point verification when extracting records for that segment.

**EXTRACT JOURNAL and Distributed Transactions**

EXTRACT JOURNAL reports on distributed transactions and supports the use of an input section recovery control file. The control file is used to complete InDoubt distributed transactions unless ALL is specified. For more information, see JCL Considerations (see page 74) as well as Common Facilities for Distributed Transactions (see page 489).
JCL Considerations

When you submit an EXTRACT JOURNAL statement through the batch command facility, in addition to the standard JCL required for the batch command facility, you must also include statements to define:

- SYS001 to point to the input archived journal file.
- SYS002 or the DDname specified in the JCL to point to the output extract file. The output extract file is a standard variable length record file. Specify a block size that is at least as large as the block size of the input journal.
- Any sort work files needed by your local sort.

To use a section recovery input control file, include a CTRLIN file definition or DD statement in the IDMSBCF execution JCL. To use a section recovery output control file, include a CTRLOUT file definition or DD statement in the IDMSBCF execution JCL. The format of both of these files is fixed block with a record length of 80.

Example

The following statement directs the EXTRACT JOURNAL utility to create an extract file from all after images in the EMPDEMO segment. The default ddname of SYS002 is used.

```
extract journal for segment empdemo all;
```

Sample Output

The first listing shown next, was generated after submitting the sample EXTRACT JOURNAL statement in the previous example. The extract file was then used as input to the ROLLFORWARD utility to restore the EMPDEMO segment. The listing from the ROLLFORWARD utility is presented after the EXTRACT JOURNAL listing.

```
EXTRACT JOURNAL listing
IDMSBCF mm/dd/yy PAGE 1
IDMS Batch Command Facility
```

Note: For considerations associated with distributed transactions during recovery operations, see the CA IDMS Database Administration Section

Note: For more information about the generic JCL used to execute the batch command facility, see the section for your operating system in this section.
AREA EMPDEMO.EMP-DEMO-REGION,
EMPDEMO.INS-DEMO-REGION,
EMPDEMO.ORG-DEMO-REGION
ALL
TO SYS002
STOP AT 'yyyy-mm-dd-hh.mm.ss.ffffff'
;
ROLLFORWARD STARTED yyyy-mm-dd-hh.mm.ss.ffffff
RU_ID 00000001 PGM_ID EMPLOAD QUIESCE LEVELS 01 UPD 00 BGIN yyyy-mm-dd-hh.
mm.ss.ffffff
RU_ID 00000001 PGM_ID EMPLOAD QUIESCE LEVELS 00 UPD 00 ENDJ yyyy-mm-dd-hh.
mm.ss.ffffff
IMAGES SELECTED FOR AREA EMPDEMO.EMP-DEMO-REGION 1,034
IMAGES SELECTED FOR AREA EMPDEMO.INS-DEMO-REGION 196
IMAGES SELECTED FOR AREA EMPDEMO.ORG-DEMO-REGION 587
TOTAL IMAGES WRITTEN 1817
JOURNAL INPUT COUNTS:
BLOCK COUNT 705 BACKWARD 0
RECORD COUNT 5526 BACKWARD 0
Status = 0
AutoCommit will COMMIT transaction
Command Facility ended with no errors or warnings

Listing from ROLLFORWARD

IDMSBCF mm/dd/yy PAGE 1

ROLLFORWARD
SEGMENT EMPDEMO
FROM EXTRACT SYS002
;
ROLLFORWARD STARTED yyyy-mm-dd-hh.mm.ss.ffffff
IMAGE FILE CREATED ON yyyy-mm-dd-hh.mm.ss.ffffff
EXTRACT FILE IMAGES CREATED FROM yyyy-mm-dd-hh.mm.ss.ffffff TO yyyy-mm-dd-hh.mm.ss.
ffffff
RECORDS RESTORED TO AREA EMPDEMO.EMP-DEMO-REGION 1,034
RECORDS RESTORED TO AREA EMPDEMO.INS-DEMO-REGION 196
RECORDS RESTORED TO AREA EMPDEMO.ORG-DEMO-REGION 587
TOTAL RECORDS RESTORED 1817
JOURNAL IMAGES READ 1,818
Status = 0
IDMSBCF mm/dd/yy PAGE 2

AutoCommit will COMMIT transaction
Command Facility ended with no errors or warnings

FASTLOAD

The FASTLOAD utility loads data into a non-SQL defined database for the first time.
This article describes the following information:

- Authorization (see page 76)
- Syntax (see page 76)
- Parameters (see page 76)
- Usage (see page 77)
- JCL Considerations (see page 84)
- Example (see page 86)
- Sample Output (see page 86)

### Authorization

<table>
<thead>
<tr>
<th>To</th>
<th>You need this privilege</th>
<th>On</th>
</tr>
</thead>
<tbody>
<tr>
<td>FASTLOAD into a segment</td>
<td>DBAWRITE</td>
<td>The segment</td>
</tr>
</tbody>
</table>

### Syntax

```
FASTLOAD
   STEP step-name
   FROM
   NOTIFY notify-record-count
   as SORTEXIT
   REUSE workfiles
```

### Parameters

- **STEP**
  Specifies that only one step of the entire FASTLOAD process should be executed. If you do not specify STEP, all steps are performed.

- **FROM**
  Specifies that FASTLOAD processing should begin at a specified step and that all remaining steps should be completed.

- **step-name**
  The name of the step to execute. The name must be one of the following:

  - SORT1
  - IDMSDBL2
  - SORT3
  - IDMSDBL3
SORT4

IDMSDBL4

NOTIFY
Directs the FASTLOAD utility to send a message to the system console after a specified number of records are read in the current step.
If you omit the NOTIFY option or specify 0, no messages are sent. Only the standard messages that go to the SYSLST file are sent when a step is completed.

notify-record-count
The number of records to read before sending a message.
If you specify zero, the only message sent is the message that is always sent to the SYSLST file at the end of each step. The message indicates the number of records processed during the step.

as SORTEXIT
Causes each DBLx step in the utility to return its input data directly from the preceding sort instead of having the sort write the data to a workfile. This option eliminates one workfile for each sort and saves the I/O it takes to write then read the workfile.

REUSE workfiles
Causes each step in the utility to reuse an existing workfile, if possible, when writing its output data, instead of writing to a new one for each step. This reduces the number of workfiles that need to be allocated.

Usage

How to submit the FASTLOAD statement
You submit the FASTLOAD statement only through the batch command facility. You must run the batch command facility in local mode.

When to use the FASTLOAD utility
Use the FASTLOAD utility to load a non-SQL-defined database for the first time.

When not to use the FASTLOAD utility
To reload a non-SQL-defined database that has been unloaded, use RELOAD.
To load an SQL-defined database, use LOAD.

Note: FASTLOAD cannot process mixed page groups and will issue an error message if mixed page groups are encountered. If your subschema binds to multiple page groups, you must select a subset of areas to process that are all in the same page group. You must use multiple invocations of the utility to process different page groups.

Before running FASTLOAD
To use the FASTLOAD utility, follow these sectionlines:

1. Write a format program to prepare input data and create a subschema that describes the records and sets to be loaded.

2. Link edit the format program with IDMSDBLU and IDMS.

3. Execute the format program.

4. Execute the FASTLOAD utility to complete the loading process.

**The format program**

The format program builds record and set ownership descriptors for each record to be loaded. These descriptors are used when loading the records.

**The subschema**

The subschema must:

- Include all records being loaded and all sets in which the records participate
- Allow the areas being loaded, and all areas with set connections to those areas, to be readied in exclusive update mode

**Run FASTLOAD all at once or in steps**

The processing initiated by the FASTLOAD utility has six steps, which you can run one at a time or as a single process. Each step generates output for use by the next step. Three of the steps are sorts to prepare the data for use by the next step. You can use your own sorting program or direct the FASTLOAD utility to sort the data during processing. If you run the steps as a single process, sorting is performed automatically.

**When to run FASTLOAD in steps**

The most common reason to use the FASTLOAD utility in steps is to cut the work into pieces, each piece requiring less time to run than the whole process.

You may also decide to use the FASTLOAD utility in steps to use your own sorting programs between the steps, or you can run the sort steps on a different machine than that holding the database.

**Restarting IDMSDBL2, IDMSDBL3 or IDMSDBL4**

If a problem arises while running IDMSDBL2 or IDMSDBL3, do not simply fix the cause of the problem and rerun the step. In addition to fixing the cause of the problem, you must do one of the following:

- Reinitialize the database and begin again at the IDMSDBL2 step
- If you backed up the database before running the step, you can run the step again, using the backup

These steps change the database, and if a problem arises, you need to undo the changes before running a step over again.
If a correctable abend occurs while running IDMSDBL4 (for example, a time-out abend), you can restart the fastload operation at the IDMSDBL4 step after unlocking the areas involved. Because the IDMSDBL4 step modifies existing records, when it is restarted it will simply modify the same records in the same way.

**SORTEXIT and FROM/STEP**

When using the FROM and STEP options with the SORTEXIT option, each pair of SORTn and DBLx steps are considered to be one step. If either half of the SORTn/DBLx is specified on a FROM or STEP option, processing will start with the SORTn step and the DBLx step will also be executed. For example:

- FROM IDMSDBL3 will start with step SORT3 and will continue to the end.
- STEP SORT3 will run steps SORT3 and IDMSDBL3.

**SORTEXIT/REUSE WORKFILE restart considerations**

Since SORTEXIT combines each SORTn step with the DBLx step that follows it, if a failure occurs in the DBLx step, a restart (if a restart is possible) must begin with the sort step and the input to the step will be resorted. Non-SORTEXIT mode will take longer to run but can be restarted after the sort in this case. Therefore, if restart time is more critical than normal runtime, do not run the utility as a sortexit.

If the REUSE WORKFILE option is used with SORTEXIT, some input workfiles will be used as output files in the same step. Therefore, if these two options are used together and a failure occurs, the utility must be restarted from the beginning.

**Workfile Considerations for restarting a failed FASTLOAD**

If the FASTLOAD command fails, depending on the reason for failure, restart the command at the failing step using the "FROM step-name" syntax. You can restart a step only if the input files to that step are intact and valid.

To prepare for a possible restart when running a one-step FASTLOAD, the Intermediate work files should have a disposition that preserves the data set in the event of an abend, for example, "DISP=(NEW,CATLG,CATLG)."

To restart FASTLOAD at a particular step, the input files to that step must have a disposition to specify that the files already exist, for example, "DISP=OLD."

To determine which files were input to a given step, refer to the "Intermediate Work File" tables under "JCL Considerations." Partially created output files should be deleted before restarting the job, and the original disposition should be used in the restart job, for example, "DISP=(NEW,CATLG,CATLG)."

The SYSPCH file contains sort parameter information for sort steps. It is an output file to IDMSDBLn steps, but is not read unless restarting or running in step mode. So during a normal run, the SYSPCH file should be treated as a normal output file, for example, "DISP=(NEW,CATLG,CATLG)." However, restarting is not as straightforward. If the previous job failed in an IDMSDBLx step, the SYSPCH file
was an output file and should be deleted before restarting. But if the failure occurred in a SORTx step, the contents of the SYSPCH file should contain the same values that were input to the SORTx step. In this case, the SYSPCH file should be preserved and defined as a SYS001 input file to the restart step.

When the SORTEXIT option is used, the SORTx and IDMSDBLx steps are combined. If a failure occurs in this mode, the SYSPCH file should normally be preserved and used as a SYS001 input file to the restart. However, there is a small window at the end of an IDMSDBLx step where the SYSPCH file is opened for output and new SORT parameters are written. If the job fails at this point, the entire SORT x/IDMSDBLx step must be restarted, but the SYSPCH file will not be valid as a SYS001 input file. In this case, the sort parameters must be recreated by hand or the job must be restarted at an earlier IDMSDBLx step if possible. One way to avoid this situation is to run in step mode when running SORTEXIT mode.

The RELDCTL data set is always an input file to the first step of a FASTLOAD whether being restarted or not.

The steps of FASTLOAD

The FASTLOAD utility consists of the following steps which you can run separately or as a single operation:

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SORT 1</td>
<td>Sorts the output file from the execution of the format program.</td>
</tr>
<tr>
<td>IDMS DBL2</td>
<td>Stores records in the database using the output from SORT1, but does not connect any sets. Creates an intermediate work file for use by SORT3. Prints statistics on the records written to the database.</td>
</tr>
<tr>
<td>SORT 3</td>
<td>Sorts the file produced by IDMSDBL2.</td>
</tr>
<tr>
<td>IDMS DBL3</td>
<td>Establishes pointers for each chained set in which each record participates (that is, builds the record prefix) but does not write the prefixes to the database. Creates an intermediate work file for use by SORT4. Builds indexes in the database.</td>
</tr>
<tr>
<td>SORT 4</td>
<td>Sorts the file produced by IDMSDBL3.</td>
</tr>
<tr>
<td>IDMS DBL4</td>
<td>Inserts the record prefixes by performing a serial sweep of the database.</td>
</tr>
</tbody>
</table>

Each step has input and output

<table>
<thead>
<tr>
<th>Step</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>SORT 1</td>
<td>SYS001 contains sort control parameters from the SYSPCH file after running the format program</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYS002 from running the format program</td>
<td>SYS004 contains the sorted contents of SYS002</td>
</tr>
<tr>
<td>IDMS DBL2</td>
<td>RELDCTL file from the format program; RELDCTL contains control and set information</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Note: Overflow statistics may be fewer than expected. To improve performance during a fastload, IDMSDBL2 uses the dbkey of the previously stored record as a 'direct' dbkey if the next record to be stored has the same new target page. This reduces the number of overflow conditions.

<table>
<thead>
<tr>
<th>Step</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>SYS001 contains sort control parameters from IDMSDBL2 if running all steps at once, no sort parameters are needed. SYS005 from IDMSDBL2</td>
<td>SYS009 contains the sorted contents of SYS005</td>
</tr>
<tr>
<td>3</td>
<td>SYS009 from SORT3 SDBL RELDCTL file from the format program; RELDCTL contains control and set information</td>
<td>SYSPCH contains sort control parameters SYS010 contains pointer descriptors</td>
</tr>
<tr>
<td>4</td>
<td>SYS001 contains sort control parameters from IDMSDBL3. If running all steps at once, no sort parameters are needed. SYS010 from IDMSDBL3</td>
<td>SYS011 contains the sorted contents of SYS010</td>
</tr>
<tr>
<td>4</td>
<td>SYS011 from SORT4 SDBL RELDCTL file from the format program; RELDCTL contains control and set information</td>
<td></td>
</tr>
</tbody>
</table>

### Note: This table describes the input and output files as if FASTLOAD were executing without the SORTEXIT and REUSE options. For the effect of these parameters, see "JCL Considerations" later in this section.

### Sort output after each step

If you run the FASTLOAD utility a step at a time, you must sort the contents of the intermediate work files. You can use your own sort program or direct the FASTLOAD utility to perform the sorts for you. If you use your own sort program, do not execute the FASTLOAD utility sort steps.

You can use the sort parameters in SYSPCH from the format programs IDMSDBL2 and IDMSDBL3 as the starting point for coding your sort parameters.

Sort the intermediate work files as follows:

<table>
<thead>
<tr>
<th>Sort name</th>
<th>File to sort</th>
<th>Sort order</th>
<th>Sort on</th>
<th>Begins at</th>
</tr>
</thead>
<tbody>
<tr>
<td>SORT1</td>
<td>SYS002</td>
<td>Descending</td>
<td>16 bytes</td>
<td>Byte 5</td>
</tr>
<tr>
<td>SORT3</td>
<td>SYS005</td>
<td>Ascending</td>
<td>16 bytes</td>
<td>Byte 5</td>
</tr>
</tbody>
</table>
### The format program

#### Functions

The format program must perform the following functions for each record to be loaded:

- Builds a record occurrence descriptor
- Builds one owner descriptor for each set in which the record is an automatic member
- Builds one owner descriptor for each set in which the record is a section member if the record is to be connected to the set at load time
- Calls IDMSDBLU, passing the record occurrence descriptor as the first argument and the owner descriptors as the remaining arguments

#### Output

IDMSDBLU uses the information provided by the format program to create an output file for use as input by the FASTLOAD utility.

#### The format program specifies the subschema, SEGMENT, and DMCL

As part of preparation for FASTLOAD operations, the format program calls IDMSDBLU. On the FIRST CALL ONLY, the subschema, segment name, and the DMCL name must be identified for use by IDMSDBLU.

For example,

```
01 PARMLIST-1.
  02 SUBSCHEMA-NAME PIC X(8) VALUE 'EMPSS01'.
  02 SEGMENT-NAME PIC X(8) VALUE 'EMPDEMO'.
  02 DMCL-NAME PIC X(8) VALUE 'EMPDMCL'.
CALL 'IDMSDBLU' USING PARMLIST-1.
```

The subschema, segment name, and DMCL must exist and be accessible at the time that the format program passes their names to IDMSDBLU, and when they are to be used by the FASTLOAD utility.

#### Record descriptors

Record occurrence descriptors built by the format program must be aligned on a doubleword and must contain the following fields:
### Field Usage Size | Description
--- | ---
1 | Char 18 bytes
   | The record name of the record being loaded. The format program must initialize this field before calling IDMSDBLU.
2 | Char 6 bytes
   | Binary zeros.
3 | Binary 4 bytes
   | Record ID of the record occurrence being loaded. The format program must initialize this field before calling IDMSDBLU.
4 | Binary 4 bytes
   | Suggested page number for storage of the record occurrence being loaded. Responsibility for supplying the page number depends on the location mode of the record:
   | If the location mode is either CALC or VIA a CALC record, IDMSDBLU determines and returns the suggested page number.
   | If the location mode is DIRECT, the format program must initialize this field either with an actual page number or with the value -1. In the latter case, IDMSDBLU returns as the suggested page number the number of the first page in the range to which the record is assigned.
   | If the location mode is VIA a VIA record, the format program must process the owner record first and save the suggested page number of the owner record to initialize this field.
5 | Binary 4 bytes
   | Binary zeros.
6 | Binary 4 bytes
   | Serial number that uniquely identifies the record occurrence being loaded. IDMSDBLU generates and returns this value.
7 | Character 4 bytes
   | Error status. IDMSDBLU returns error-status codes that parallel those returned by CA IDMS/DB. The format program must ensure that the returned value is 0000 before proceeding.
8 | Character
   | Size of the largest record to be loaded
   | Actual record occurrence, left justified. The format program must initialize this field before calling IDMSDBLU.

After the last record is passed to IDMSDBLU, it is necessary to call IDMSDBLU one final time. This call must have the RECORD descriptor with the record ID field set to a -1 as its parameter.

### Owner descriptors

Owner descriptors built by the format program must be aligned on a fullword and must contain the following fields:

<table>
<thead>
<tr>
<th>Field Usage</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1 | Character 16 bytes
   | Name of the set in which the record being loaded is a member, left justified. The format program must initialize this field before calling IDMSDBLU. |
| 2 | Binary 4 bytes
   | Serial number of the owner record. The format program must initialize this field only if the owner record has a location mode of CALC and duplicates are allowed. In this case, the format program must process the owner record first and save the returned serial number (field 4 of the record occurrence descriptor) to
Field Usage  | Size  | Description
---|---|---
| initialize the field. The format program does not need to initialize this field for system-owned indexes.
3  | CALC | 1 through 256 bytes CALC key of the owner record, left justified. The format program must initialize this field before calling IDMSDBLU.
4  | Serial number of the owner record. In this case, the format program must process the owner record first and save the returned serial number (field 4 of the record occurrence descriptor) to initialize the field. The format program does not need to initialize this field for system-owned indexes.

JCL Considerations

When you submit a FASTLOAD utility through the batch command facility, the JCL to execute the facility must include statements to define:

- Files containing the areas to be loaded
- The intermediate work files
- Sort space

⚠️ Note: For more information about the generic JCL used to execute the batch command facility, see the section for your operating system in this section.

Intermediate work files

The following tables indicate which work files are created and read by the different utility steps depending on the use of the SORTEXIT and REUSE WORKFILE options.

<table>
<thead>
<tr>
<th>Step</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>FASTLOAD: NOT sortexit mode and NOT reusing workfiles</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SORT1</td>
<td>SYS002</td>
<td>SYS004</td>
</tr>
<tr>
<td>IDMSDBL2</td>
<td>SYS004</td>
<td>SYS005</td>
</tr>
<tr>
<td>SORT3</td>
<td>SYS005</td>
<td>SYS009</td>
</tr>
<tr>
<td>IDMSDBL3</td>
<td>SYS009</td>
<td>SYS010</td>
</tr>
<tr>
<td>SORT4</td>
<td>SYS010</td>
<td>SYS011</td>
</tr>
<tr>
<td>IDMSDBL4</td>
<td>SYS011</td>
<td></td>
</tr>
</tbody>
</table>
### Work file JCL Considerations for STEP mode

FASTLOAD normally runs as a single step but runs as separate steps using the "STEP step-name" syntax. When running in step mode, input files should have dispositions to state that the file already exists, for example, "DISP=OLD."

Preserve output files on successful completion but not when the job fails, for example, "DISP=(NEW, CATLG,DELETE)."

See the "Intermediate Work File" table to determine which files are input and which files are output and when they are used.

The RELDCTL file is always input to every step.

The SYSPCH file is created by an IDMSDBLx step and used as input to a SORTx step. When used as input, it is defined as SYS001.

Work file record lengths:

- The RELDCTL file is a fixed length file with a record length of 60 bytes.
- The SYSPCH file is a fixed length file with a record length of 80 bytes.
All SYSxxx files are variable length files. The record length can vary from one step to the next, from one job to the next. Do not code an LRECL value in the JCL, just code a BLKSIZE value. A BLKSIZE value should be chosen based on the optimal size for the device being used, for example, 1/2 track if disk or 32k if tape.

Example

The following example directs the FASTLOAD utility to perform an initial load of a sample CA IDMS /DB non-SQL-defined database.

```
fastload;
```

The following command directs FASTLOAD to run all steps as a sortexit and to reuse workfiles:

```
fastload as sortexit reuse workfiles;
```

Sample Output

After successful completion of the FASTLOAD utility, the CA IDMS Batch Command Facility produces the following listing:

```
IDMSBCF mm/dd/yy PAGE 1
IDMS Batch Command Facility

SET BATCH WIDTH PAGE 80;
Status = 0

FASTLOAD;
UT010002 BEGINNING PROCESSING FOR STEP SORT1
UT009001 IDMSDBLY RELEASE nn.n TAPE volser SORT STARTED
UT009002 12 RECORDS WERE READ FROM SYS002
UT009003 12 RECORDS WERE WRITTEN TO SYS004
UT009004 IDMSDBLY RELEASE nn.n SORT COMPLETED SUCCESSFULLY
UT010003 STEP SORT1 HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP IDMSDBL2
UT005001 IDMSDBL2 RELEASE nn.n TAPE volser PROCESSING STARTED
DATABASE LOAD STATISTICS
DATABASE LOADED ON mm/dd/77 AT 154852
PAGES READ .............. 4
PAGES WRITTEN ............ 5
PAGES REQUESTED .......... 7
CALC RCDs IN TARGET PAGE . 5
CALC RCDs OVERFLOWED ...... 1
VIA RCDs IN TARGET PAGE .. 3
VIA RCDs OVERFLOWED ...... 0
LINES REQUESTED BY IDMS .. 26
RCDs MADE CURRENT OF R/U . 9
CALLs TO IDMS .......... 13
FRAGMENTS STORED ........ 0
RECORDs RELOCATED ......... 0
UT005003 12 INTERMEDIATE RECORDs WERE WRITTEN TO SYS005
UT005004 SYS005 RECORD LENGTH IS 44
UT005005 NO DATABASE ERRORS WERE ENCOUNTERED
UT005006 IDMSDBL2 RELEASE nn.n PROCESSING COMPLETED
UT010003 STEP IDMSDBL2 HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP SORT3
UT009001 IDMSDBLY RELEASE nn.n TAPE volser SORT STARTED
UT009002 12 RECORDS WERE READ FROM SYS005
UT009003 12 RECORDS WERE WRITTEN TO SYS009
UT009004 IDMSDBLY RELEASE nn.n SORT COMPLETED SUCCESSFULLY
UT010003 STEP SORT3 HAS COMPLETED SUCCESSFULLY
```
CA IDMS Reference - 19.0

UT010002 BEGINNING PROCESSING FOR STEP IDMSDBL3
UT006001 IDMSDBL3 RELEASE nn.n TAPE volser PROCESSING STARTED
UT006007 12 INTERMEDIATE RECORDS WERE READ FROM SYS009
UT006002 12 INTERMEDIATE RECORDS WERE WRITTEN TO SYS010
UT006005 NO DATABASE ERRORS WERE ENCOUNTERED
UT006006 IDMSDBL3 RELEASE nn.n PROCESSING COMPLETED
UT010003 STEP IDMSDBL3 HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP SORT4
UT009001 IDMSDBLY RELEASE nn.n TAPE volser SORT STARTED
UT009002 12 RECORDS WERE READ FROM SYS010
UT009003 12 RECORDS WERE WRITTEN TO SYS011
UT009004 IDMSDBLY RELEASE nn.n SORT COMPLETED SUCCESSFULLY
UT010003 STEP SORT4 HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP IDMSDBL4
UT007001 IDMSDBL4 RELEASE nn.n TAPE volser PROCESSING STARTED
UT007004 NO DATABASE ERRORS WERE ENCOUNTERED
UT007005 NO LOGIC ERRORS WERE ENCOUNTERED
UT007002 12 RECORDS WERE READ FROM SYS011
UT007006 IDMSDBL4 RELEASE nn.n PROCESSING COMPLETED
UT010003 STEP IDMSDBL4 HAS COMPLETED SUCCESSFULLY
UT010001 DATABASE LOAD HAS COMPLETED SUCCESSFULLY
Status = 0

AutoCommit will COMMIT transaction

Command Facility ended with no errors or warnings

![Note: For more information about loading a non-SQL defined database see the CA IDMS Database Administration Section.]

FIX ARCHIVE

The FIX ARCHIVE utility rewrites a tape journal file. Typically, you rewrite a tape journal file in use at the time of an abnormal system shutdown to make the file usable by the ROLLBACK utility.

This article describes the following information:

- Authorization (see page 87)
- Syntax (see page 88)
- Parameters (see page 88)
- Usage (see page 88)
- JCL Considerations (see page 89)
- Examples (see page 90)
- Sample Output (see page 90)

Authorization

| To Rewrite a tape journal file | You need this privilege | On The DMCL |
Syntax

```
FIX ARCHIVE └── REPORT ┌── FULL ┌── TERSE └── SUMMARY
```

Parameters

- **REPORT**
  Specifies the amount of detail that is to appear on the report.

- **FULL**
  Specifies that all details are to be reported. This includes for every transaction: checkpoints, database statistics, and area usage. All details of a distributed transaction record are reported. This includes local transaction ids with program names; external transaction ids, and resource manager interests. Additionally, transactions active at the end of the process are listed and the time of the last global quiesce point is identified. FULL is the default if no REPORT option is specified.

- **TERSE**
  Indicates that only transaction checkpoints and summary information is produced. For distributed transaction records: external transaction ids, and resource manager interests are not included in the report.

- **SUMMARY**
  Indicates that only final summary information is produced.

Usage

**How to submit the FIX ARCHIVE statement**

You submit the FIX ARCHIVE statement to CA IDMS/DB only through the batch command facility. You must run the batch command facility in local mode.

**How CA IDMS/DB rewrites a tape journal file**

When rewriting a tape journal file, CA IDMS/DB:

- Writes an end-of-file marker on the output tape
- Writes ABRT checkpoints for all run units active at the time of the abnormal termination
- Prints checkpoints and program statistics for run units whose activity is recorded in the journal file
- Identifies quiesce points for use in a rollback or rollforward operation
- Writes additional checkpoint records to the output tape
For incomplete distributed transactions whose state is InDoubt at the time of the abnormal termination, additional distributed and local checkpoint records will be written to the output file to complete the transaction if a matching section recovery control input file entry is provided.

For local (that is, non-distributed) transactions still active at the time of the abnormal termination, an ABRT checkpoint record will be added to the output file.

Multivolume journal files

The FIX ARCHIVE statement must process all unarchived volumes of a tape journal file in a single run.

Archived journal files

In most cases, you do not have to run the FIX ARCHIVE utility statement against archived journal files. Here are some exceptions:

- To identify incomplete distributed transactions whose state is InDoubt at the end of the input file. (PRINT JOURNAL can also be used for this purpose.)
- To complete transactions by writing additional checkpoint records to the output file.
  - For incomplete distributed transactions whose state is InDoubt at the end of the input file, additional distributed and local checkpoint records will be written to the output file if a matching section recovery control input file entry is provided.
  - For local (that is, non-distributed) transactions still active at the time of the abnormal termination, an ABRT checkpoint record will be written.
- To merge multiple journal tapes onto one tape for certain rollback and rollforward operations that require multiple journal tapes to be one contiguous file

FIX ARCHIVE and Distributed Transactions

FIX ARCHIVE reports on distributed transactions and supports the use of input and output section recovery control files. The input section recovery control file is used to complete InDoubt distributed transactions. If an output section recovery control file is included in the JCL, an entry will be written for each incomplete distributed transaction encountered. For more information, see JCL Considerations (see page 89) and the "Common Facilities for Distributed Transactions" section.

Note: For considerations associated with distributed transactions during recovery operations, see the CA IDMS Database Administration Section.

JCL Considerations

When you submit a FIX ARCHIVE statement to CA IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define:
- The tape journal file to be rewritten, which is on SYS001
- The rewritten tape journal file, which is on SYS002

To use a section recovery input control file, include a CTRLIN file definition or DD statement in the IDMSBCF execution JCL. To use a section recovery output control file, include a CTRLOUT file definition or DD statement in the IDMSBCF execution JCL. The format of both of these files is fixed block with a record length of 80.

⚠️ **Note:** For more information about the generic JCL used to execute the batch command facility, see the section for your operating system in this section.

### Examples

The following statement directs the FIX ARCHIVE statement to rewrite a tape journal file.

```plaintext
fix archive;
```

### Sample Output

When the FIX ARCHIVE utility runs successfully, the following listing is produced:

```
IDMSBCF IDMS Batch Command Facility mm/dd/yy PAGE 1

FIX ARCHIVE;
RU_ID 1 PGM_ID EMPLOAD QUIESECE LEVELS 1 UPD 0 BEGIN yyyy-mm-dd-hh.mm.ss.ffffff
RU_ID 1 PGM_ID EMPLOAD QUIESECE LEVELS 0 UPD 0 ENDJ yyyy-mm-dd-hh.mm.ss.ffffff

STATISTICS FOR EMPLOAD RU_ID 1

PAGES READ 969 PAGES WRITTEN 847 PAGES REQUESTED 2567 CALC TARGET 186
CALC OVERFLOW 0 VIA TARGET 439 VIA OVERFLOW 0 LINES REQUESTED 6842
RECS CURRENT 1307 CALLS TO IDMS 1461 FRAGMENTS STORED 0 RECS LOCATED 0
LOCKS REQUESTED 0 SELECT LOCKS 0 UPDATE LOCKS 0

START TIME: 1999-09-18-15.52.35.481748 TIME OF LAST COMMIT: NONE NUMBER OF COMMITS: 0

SINCE LAST COMMIT

<table>
<thead>
<tr>
<th>TABLESPACES OPENED</th>
<th>BEFORE</th>
<th>AFTER</th>
<th>SINCE LAST COMMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPDEMO.ORG-DEMO-REGION</td>
<td>825</td>
<td>825</td>
<td>825</td>
</tr>
<tr>
<td>EMPDEMO.INS-DEMO-REGION</td>
<td>115</td>
<td>115</td>
<td>115</td>
</tr>
<tr>
<td>EMPDEMO.EMP-DEMO-REGION</td>
<td>1314</td>
<td>1314</td>
<td>1314</td>
</tr>
</tbody>
</table>

ACTIVE PROGRAMS AT STOP TIME WERE:
NONE

DATABASE MAY NOT NEED TO BE RECOVERED
Status = 0

AutoCommit will COMMIT transaction

Command Facility ended with no errors or warnings
```
Note: For more information about journaling, see the CA IDMS Database Administration Section.

FIX PAGE

The FIX PAGE utility verifies, and optionally modifies, the contents of a database page.

This article describes the following information:

- Authorization (see page 91)
- Syntax (see page 91)
- Parameters (see page 92)
- Usage (see page 92)
- JCL Considerations (see page 93)
- Examples (see page 93)
- Sample Output (see page 94)

Authorization

<table>
<thead>
<tr>
<th>To</th>
<th>You Need This Privilege</th>
<th>On</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modify a page in an area</td>
<td>DBAWRITE</td>
<td>The area</td>
</tr>
<tr>
<td>Verify a page in an area</td>
<td>DBAWRITE</td>
<td>The area</td>
</tr>
</tbody>
</table>

Syntax

```
FIX

ALREADY LOCKED

PAGE page-group: page-number

VERIFY REPlace hex-offset hex-halfword
```
Parameters

- **ALREADY LOCKED**
  Specifies that if the target area or areas are currently locked, processing will continue. This option is honored in local mode only. It is ignored when processing under the central version and has no effect on the processing of unlocked areas.

- **PAGE**
  Specifies the page containing the data to be verified or replaced.

- **page-group**:
  The page group of the area containing the page you want to verify or replace.
  If you do not specify a page group, page group zero is used.

- **page-number**
  The number of a page in an area included in the DMCL module.

- **VERify**
  Directs the FIX PAGE utility to verify that the data starting at the specified offset is equal to the specified halfwords.

- **REPlace**
  Directs the FIX PAGE utility to replace the data starting at the specified offset with the specified halfwords.

- **hex-offset**
  A 4-digit hexadecimal offset from the beginning of the page.
  The offset identifies the starting position of the data to be verified or replaced. An offset of 0000 indicates that the data begins with the first byte on the page.

- **hex-halfword**
  A 4-digit hexadecimal value representing two bytes of data (one halfword) to be used:

  - **For comparison to the data on the page**, when *hex-halfword* occurs in a VERIFY parameter
  - **As a replacement for data on the page**, when *hex-halfword* occurs in a REPLACE parameter

  You can specify up to 19 halfwords in a single VERIFY or REPLACE parameter. Multiple halfwords must be separated by commas.

Usage

How to submit the FIX PAGE statement

You submit the FIX PAGE statement by using either the batch command facility or the online command facility.

If a VERIFY fails
If you specify VERIFY and REPLACE, CA IDMS/DB verifies existing data before making any modifications to the contents of a page. If any of the data supplied in the VERIFY parameter do not match the existing contents of the page, data replacements are not made on the page, and the FIX PAGE operation terminates with an error.

**Repairing a locked area**

If a local mode application abends while an area is being updated, the lock could remain on the area. In this case, you can either explicitly unlock the area using the UNLOCK utility, or you can use the ALREADY LOCKED option.

If ALREADY LOCKED is specified and the area was locked, the area will remain locked after the FIX PAGE is completed. The ALREADY LOCKED option is not required if a page is being fixed online or through batch/CV and is ignored if specified.

**Unlocking a locked area**

The FIX PAGE utility cannot be used to update an area's physical area lock. Instead, use the LOCK and UNLOCK area utility statements to do this.

**Committing prior work**

Before executing this utility under a central version, you must commit any previous work done within the current session.

**JCL Considerations**

When you submit a FIX PAGE statement to CA IDMS/DB through the batch command facility in local mode, the JCL to execute the facility must include statements to define the files containing the pages to be processed. To run the batch command facility under central version, include a SYSCTL statement.

⚠️ **Note:** For more information about the generic JCL used to execute the batch command facility, see the section for your operating system in this section.

**Examples**

**Verifying page contents**

The following FIX PAGE statement requests verification of four bytes of data at the hexadecimal offset 0030 and two bytes of data at the hexadecimal offset 0048 on page 75,003:

```plaintext
fix page 75003
  verify 0030 0125,5F0F
  verify 0048 7822;
```

**Replacing data on a single page**
The following FIX PAGE statement verifies and replaces data at one offset on page 75,020. If the data being verified is incorrect, FIX PAGE does not replace any data on the page.

```
fix page 75020
   verify 0066 C1D9,D440
   replace 0066 D3C5,C740;
```

**Replacing data on multiple pages**

The following FIX PAGE statement verifies and replaces data on three pages. FIX PAGE replaces the data on each specified page only if all the data being verified on the page is correct.

```
fix page 224521
   verify 0200 89EE,F2C3
   replace 0200 89E5
page 263942
   verify 00C2 440A,1254,339B
   verify 0110 5B2A,872F,AA23
   replace 00C2 3F24,85D2,1087
   replace 0110 5B24,8733,2842
page 263957
   verify 0124 8924,3258
   replace 0124 0000,3268;
```

**Sample Output**

The following listing was generated after successfully replacing data on page 75020 in example two.
FORMAT

The FORMAT utility prepares a database file, area, segment, SYSTRK or disk journal file for use by CA IDMS/DB.

This article describes the following information:

- Authorization (see page 95)
- Syntax (see page 95)
- Parameters (see page 96)
- Usage (see page 98)
- JCL Considerations (see page 103)
- Examples (see page 104)
- Sample Output (see page 105)

Authorization

To FORMAT You Need This Privilege On
An area DBAWRITE The area
A segment DBAWRITE All areas of the segment
A file DBAWRITE All areas that map to the file
A disk journal file USE The DMCL
A SYSTRK file USE The DMCL

Syntax

```
FORMAT

SEGMENT segment-name
FILE segment-name.file-name
AREA segment-name.area-name
JOURNAL journal-file-name
SYSTRK SYSTRK-format-options
```

Expansion of SYSTRK-format-options
Parameters

- **SEGMENT**
  Formats all files associated with the specified segment.

- **segment-name**
  Specifies the name of a segment included in the DMCL module.
  
  ![](Note: If you are formatting a segment of an SQL-defined database, synchronization time stamps for all areas in the segment are also stored.

- **FILE**
  Formats the specified file.

- **segment-name**
  Identifies the segment associated with the file.

- **file-name**
  Identifies the name of a database file described by the DMCL module.

- **AREA**
  Formats the files associated with the specified area. Only the portion of each file that maps to pages in the area is formatted.
  
  ![](Note: You can use the AREA parameter only when you are reformatting an area that has been formatted at least once before.

- **ALREADY LOCKED**
  Specifies that if the target area or areas are currently locked, processing will continue. This option is honored in local mode only. It is ignored when processing under the central version and has no effect on the processing of unlocked areas.
- **segment-name**
  Specifies the name of the segment associated with the area.

- **area-name**
  Specifies the name of an area included in the DMCL module.

- **JOURNAL**
  Formats the specified disk journal file.

- **journal-file-name**
  Specifies the name of a disk journal file included in the DMCL module.

- **ALL**
  Formats all disk journals in the DMCL.

- **MAX AREA nnn**
  The maximum number of areas to define for the journal, where nnn is an integer from 1 to 100,000. The actual number of areas that CA IDMS/DB can handle can be higher because of rounding and the size of a journal block.

- **DATA STORAGE SIZE nnn**
  Specifies the amount of space to reserve in 1K (1024 bytes) increments for Data Storage in a journal file, where nnn is an integer from 1 32,767.

- **FAST**
  Formats only the journal header blocks of already existing and formatted journal files. If MAX AREA is specified, the number of JHDA entries are recalculated and the number formatted may change. If the STORAGE clause is specified, the number of JHD2 entries are recalculated and the number formatted may change.

- **SYSTRK**
  Indicates that one or more SYSTRK files are to be formatted.

- **target-ddn**
  Specifies the DDname or linkname of a SYSTRK file to be formatted.

⚠️ **Note:** The target-ddn should not begin with the DDname prefix used for referencing SYSTRK files. Otherwise, CA IDMS attempts to use it for building the DMCL definition and fails if it cannot do so.

- **INITIAL**
  Indicates this is the first time the SYSTRK file is being formatted. If specified, no check is made to determine whether the file is in use by a CV. INITIAL must be specified the first time a file is formatted. It should not be specified when a file is being re-formatted unless you are sure that the file is not in use by a CV.

- **file-cnt**
  Specifies the number of SYSTRK files that are maintained as active mirrors. The file-cnt must be an integer in the range 2 through 4. If file-cnt is not specified, the value is taken from the file identified by ddname2, if specified, or at run time from the file count currently in use by the DC/UCF system. If file-cnt has never been
specified on a format for a related SYSTRK file, the first time that a DC/UCF system writes to a set of SYSTRK files, it sets file-cnt to be the lesser of the number of files currently allocated and 4. The value can be altered dynamically by a DCMT VARY CHANGE TRACKING command.

- **DELETE**
  Specifies whether the DC/UCF system is to automatically delete obsolete SYSTRK files.
  - **ON**
    (z/OS and z/VM systems only) Enables automatic file deletion.
  - **OFF**
    Disables automatic file deletion.

If DELETE is not specified, the value is taken from the file identified by ddname2, if specified, or at run time from the delete option currently in use by the DC/UCF system. If DELETE has never been specified on a format for a related SYSTRK file, the first time that a DC/UCF system writes to a set of SYSTRK files, it sets the option to OFF. The option can be altered dynamically by a DCMT VARY CHANGE TRACKING command.

- **page-size**
  Specifies the size of each page to be written to the SYSTRK file being formatted and must be an integer in the range 4088 through 32764. On z/VM, the value must be 4096. On z/OS, the maximum page-size is 32760. Do not specify page-size for VSAM files.

- **number-of-pages**
  Specifies the number of pages to be written to the SYSTRK file being formatted and must be an integer in the range 10 through 999,999.

- **ddname2**
  Specifies the DDname of a file from which the attributes and contents may be taken. If ddname2 is specified together with either or both page-size and number-of-pages, the latter values override the respective attributes of the file identified by ddname2.

  **Note:** ddname2 should *not* begin with the DDname prefix used for referencing SYSTRK files unless the identified file contains the DMCL definition to be used during execution of the command facility.

- **ACTIVE**
  Indicates that the contents of the file identified by ddname2 are to be copied to the file being formatted even if the file identified by ddname2 is currently in-use by a CV.

- **pct-increase**
  Specifies the percentage increase in the number of pages written to the file being formatted over the number of pages in the file identified by ddname2. The pct-increase must be an integer in the range 0 through 1000.

---

**Usage**

**How to submit the FORMAT statement**
A FORMAT AREA or FORMAT SEGMENT statement can be submitted through either the batch command facility or the online command facility. The batch command facility can be run in either local mode or under the central version. When submitted in local mode, FORMAT SEGMENT will behave like a FORMAT FILE for each file in the segment. When running under central version, it will behave like a FORMAT AREA for each area in the segment.

Note: The FORMAT FILE, FORMAT SYSTRK, and FORMAT JOURNAL statements can be submitted only through the batch command facility in local mode.

When to use FORMAT

You must use the FORMAT utility to prepare:

- Database files before loading any data into the database
- Disk journal files before any journaling to the files occurs
- SYSTRK files before they can be referenced as SYSTRK files in any CA IDMS execution JCL.

When necessary, you also use the FORMAT statement to reformat database files and areas, SYSTRK files and disk journal files.

When not to use the FORMAT statement

Do not try to reformat a file, area, or journal that is currently active under a DC/UCF system. Do not format an existing journal if it contains journal records that might be needed for recovery.

When formatting SQL-defined segments

When formatting an area or segment that contains SQL-defined tables, be sure you are connected to the catalog segment where the table definitions reside. You can issue an explicit CONNECT to the DBNAME of the dictionary containing the catalog segment or to the actual segment where the table definitions reside.

How FORMAT formats a database file or area

The FORMAT statement formats a database file or area into pages using information contained in the DMCL module. When formatting a database file or area, the FORMAT statement:

- Establishes space management pages (SMPs)
- Initializes the space management entry for each database page
- Establishes a header and footer on each database page
- Sets all data portions of each database page to binary zeros
- Stores synchronization stamps when formatting areas or segments of an SQL-defined database
Restriction on the AREA parameter

When formatting a database by file, a sequential access method (QSAM) is used. When formatting a
database by area, a direct access method is used. Because the database must be formatted into
blocks by a sequential access method before it can be processed using a direct access method, you
can use the AREA parameter of the FORMAT statement only when you are reformating the area.

A format by segment in local mode is equivalent to format by file in this regard, and can be used to
format new files. Format by segment under central version is equivalent to format by area and can
only be used to reformat existing areas.

Area format depends on the area lock

In local mode, to prevent inadvertent formatting of an area that is being updated by another
application, the area is locked for the duration of the operation when formatting by area. If the area
is already locked, the format will not take place.

Formatting a locked area

If a local mode application abends while an area is being updated, the lock could remain on the area.
In this case, you can either explicitly unlock the area using the UNLOCK utility, you can format by file,
or you can use the ALREADY LOCKED option.

If ALREADY LOCKED is specified and the area was locked, the area will remain locked after the format
is complete. The ALREADY LOCKED option is not required if formatting an area online or through
batch/CV and is ignored if specified.

Formatting by segment does not lock areas

If you format by segment in local mode, the FORMAT utility does not lock the areas involved.

Reformatting an area

You can reformat a database by file, by area, or by segment. Processing by file is more efficient than
processing by area. However, if you format by file in an SQL-defined database, you must run the
INSTALL STAMPS utility for each affected area or segment.

How FORMAT formats a disk journal file

The FORMAT utility formats a disk journal file into blocks according to the journal file definition in a
DMCL module. The FORMAT utility writes a journal header record at the beginning of each block and
sets the remainder of the block to binary zeros, except when the FAST parameter is specified. The
FAST parameter, used only with previously initialized journal files, reinitializes the header files only.

Formatting disk journal files

A certain amount of space is reserved in each disk journal file for information about other systems
with which a system communicates. All journal files must have the same amount of space since the
data in one journal file is replicated to every other journal file.
You can specify the size or allow it to default. The actual size allocated may be higher than the value specified due to rounding. Space is allocated in blocks whose size is (journal block size minus 256). By default, one block is allocated. Additional blocks are allocated if needed until the total size meets or exceeds the size specified. If the journal block size is less than 256, no space will be reserved.

Committing prior work

Before executing this utility under a central version, you must commit any previous work done within the current session. For more information, see Central Version Considerations (see page 24).

When to use MAX AREA

Normally when a journal is formatted, a fixed number of JHDA blocks are created. A JHDA block stores the ready status of areas for warmstart purposes. The size of a journal block and the number of JHDAs limit the number of areas that a Central Version can have open at one time.

The MAX AREA option lets a journal be formatted that can handle more areas without increasing the size of a journal block. Ideally, the size of a journal block should be optimized to improve runtime efficiency, and should not be affected by the number of areas that might exist.

The MAX AREA option can also reduce the number of JHDA blocks that are created, which frees journal space.

To calculate the number of areas that one JHDA journal block can handle, use the following formula:

\[ \text{number of AREAS} = \frac{(\text{jrnl blksize} - 32)}{8} \]

When MAX AREA is not specified, the FORMAT utility command will create 3 JHDA blocks. The number of areas that the default will support will vary depending on the journal block size. For example if the journal were formatted with a block size of 2032 bytes, 750 areas could be open at one time, 250 per JHDA.

Native VSAM files

Native VSAM files that are to be accessed by CA IDMS/DB are not structured like CA IDMS/DB database files. Do not use the FORMAT statement against native VSAM files.

Formatting areas and segments under central version

The areas to be formatted must be in update mode to CV. The physical area lock for each area will remain on during and after the format. A logical area lock is acquired to prevent online transactions from updating the area during the format. If other online transactions are holding a logical area lock, Format will wait until the locks are released. If the wait would cause a deadlock, the format will be aborted. Once acquired, the logical area lock is released when a commit transaction is explicitly issued, or implicitly when the batch step ends, or at the end of the pseudo-converse when running online.

Referencing SYSTRK Files During Format

To avoid I/O errors when building the runtime environment in local mode, only previously formatted files should be referenced using a DDname that matches the SYSTRK DDname prefix. For this reason, it is recommended that non-matching DDnames always be used to identify SYSTRK files being formatted.
SYSTRK File Attributes

If a SYSTRK file is being formatted to be added as a mirror of an existing file, the page sizes of the two files must be the same and the file being formatted must have at least as many pages as the existing file. If these criteria are not met the following conditions can occur:

- Any attempt to make the newly formatted file an active mirror of the existing file fails.
- If a LIKE parameter is specified, FORMAT does not copy the contents of the file specified by ddname2 to the newly formatted file.

If INITIAL is not specified, the page size and number of pages of a file being formatted remain unchanged.

If INITIAL is specified, the number of pages written to the file is determined according to the following precedence rules:

- If a FILE SIZE parameter is specified, then the number of pages is number-of-pages.
- If a LIKE parameter is specified, then the number of pages is a value based on the number of pages in the file identified by ddname2. The value is calculated as:
  \[
  \text{page-cnt} \times \frac{(100 + \text{pct-increase})}{100}
  \]
  - page-ctnt
    Specifies the number of pages in the file identified by ddname2.
  - pct-increase
    Specifies the value in the EXPAND parameter, if specified or 0.

- The number of pages is a value based on the size of the current DMCL calculated as:
  \[
  \frac{(\text{DMCL-size} + \text{page-size -1})}{\text{page-size}} \times 4
  \]
  - DMCL-size
    Specifies the size of the DMCL load module.
  - page-size
    Specifies the page size of the file being formatted.

In the latter two cases, the number calculated is rounded up to the next larger integer value. If the calculated value is less than the minimum, it is set to the minimum of 10. If the calculated value is larger than the maximum, it is set to the maximum of 999,999.

If INITIAL is specified, the size of the pages written to a non-VSAM file is determined according to the following precedence rules:

- If PAGE SIZE is specified, then the page size is page-size.
- If a block size has been assigned (for example, specified in JCL or at the time the file was created), then page size is the block size.
- If a LIKE parameter is specified, then page size is the page-size of the file identified by ddname2.
Choosing a SYSTRK Page Size

In most cases, the FORMAT utility's default page size for SYSTRK files provides an acceptable trade-off between memory, I/O, and disk space. Consider overriding the default only if the size of the DMCL is extremely large (500K or more). A larger page size will reduce I/Os and disk space requirements at the expense of slightly increased memory usage for buffers.

Estimating the Minimum Number of Pages for a SYSTRK File

To estimate the minimum number of pages needed for a SYSTRK file, perform the following steps:

1. Take the size of the DMCL load module used by the CV, divide it by the SYSTRK page size and multiply it by 2.5.

2. Multiply the resulting value with a factor to allow for overrides and growth. Overrides require approximately 100 bytes of space each and are generated for:
   - Each database or journal file defined in the execution JCL
   - Each dynamic change in the data set name of a database or journal file
   - Each dynamic change in the permanent status of an area
   - Each dynamic change in the status of a journal file

Copying SYSTRK File Contents

If a LIKE parameter is specified, the contents of the file identified by `ddname2` are copied to the files being formatted unless the file identified by `ddname2` is in use by a CV or the attributes of the two files are incompatible. If the contents of `ddname2` are not copied, a message indicates the reason.

If the file attributes are compatible, specify the keyword ACTIVE to force the copy to occur even if the file identified by `ddname2` is in use by CV. Only do this if you are sure that CV will not update the file while the copy is in progress, otherwise, the contents of the two files may not be the same which can lead to unpredictable results during CV restart. Ensure that a CV does not update its SYSTRK files by varying change tracking inactive before doing the format.

There is normally no need to force the contents of SYSTRK files to be copied. CV automatically updates newly formatted SYSTRK files as part of making them active mirrors.

JCL Considerations

When you submit a FORMAT statement through the batch command facility in local mode, the JCL to execute the facility must include statements to define the following:

- Database files to be processed (or that map to the areas to be processed)
Journal files to be processed

Dictionary containing table definitions, if formatting all or part of an SQL-defined database by area or by segment.

SYSTRK files to be processed

To run under the batch command facility under central version, include a SYSCTL statement.

⚠️ Note: For more information about the generic JCL used to execute the batch command facility, see the section for your operating system in this section.

Examples

Formatting a segment

The following illustration shows a segment that includes three files:

```
+-----------------+-----------------+-----------------+
| FILE_1 | FILE_2 | FILE_3 |
+-----------------+-----------------+-----------------+
    | SEGMENT_A     |
+-----------------+-----------------+-----------------+
```

To format all the files associated with the segment illustrated above, you could use the following FORMAT statement:

```
format segment segment_a;
```

Reformatting by database file

The following left-hand illustration shows two areas, each of which maps to a single database file. The right-hand illustration shows a single area that maps to two database files.

```
+-----------------+-----------------+-----------------+
| FILE_1 | FILE_2 |
+-----------------+-----------------+-----------------+
    | AREA_A | AREA_B |
+-----------------+-----------------+-----------------+
```

The FORMAT utility can reformat the portions of the database illustrated above either by file or by area. However, because processing by file is more efficient, you would use the following FORMAT statements to do the job in both cases:

```
format file segment_a.file_1;
format file segment_a.file_2;
```

Reformatting by area

The following left-hand illustration shows two areas that map to a single database file. The right-hand illustration shows an area that maps to two database files, one of which also contains another area.

```
+-----------------+-----------------+
| FILE_1 |
+-----------------+-----------------+
```

```
+-----------------+-----------------+
| FILE_1 | FILE_2 |
+-----------------+-----------------+
    | AREA_A     |
+-----------------+-----------------+
```
To reformat AREA_A in either illustration, you must use the following FORMAT statement:
format area segment_a.area_a;

Formatting a journal file

The following FORMAT statement requests formatting of the disk journal named SYSJRN1:
format journal sysjrnl1;

Formatting by SEGMENT

In the following example, three database files are formatted for an SQL-defined database using the SEGMENT option of the FORMAT utility.

A CONNECT to the segment or database name containing the table definitions is issued. This is necessary so that the INSTALL STAMPS utility can automatically install area and table stamps during the FORMAT operation.

connect to syssql;
 format segment userdb;

Formatting SYSTRK Files

The following sample IDMSBCF statement instructs the FORMAT utility to format three new SYSTRK files (track01, track02, track03). It directs the utility to format the files to have the default page size of 7548 and contain 60 pages each. CA IDMS will maintain 3 active mirrors.

format systrk track01, track02, track03
  initial file count 3
  file size 60;

Sample Output

Formatting by SEGMENT

The following listing was generated after the successful completion of the previous FORMAT SEGMENT USERDB example.

IDMSBCF  IDMS Batch Command Facility
mm/dd/yy   PAGE 5

CONNECT TO SYSSQL;
Status = 0
FORMAT SEGMENT USERDB;

File USERDB.EMPF1 blocks 1 to 50.
Area USERDB.EMP_AREA pages 5,001 to 5,050.
Page size in file 4,096.

File USERDB.ORGF1 blocks 1 to 50.
Area USERDB.ORG_AREA pages 5,051 to 5,100.
Page size in file 4,096.

File USERDB.EMPIX blocks 1 to 50.
Area USERDB.EMPX_AREA pages 5,101 to 5,150.
Page size in file 2,000.
Status = 0
AutoCommit will COMMIT transaction
Command Facility ended with no errors or warnings

⚠️ Note: For more information about allocating CA IDMS/DB files, see the CA IDMS Database Administration Section.

Formatting SYSTRK Files

The following listing was generated after the successful completion of the previous FORMAT SYSTRK example.

IDMSBCF CA IDMS Batch Command Facility

```
FORMAT SYSTRK TRACK01, TRACK02, TRACK03
  INITIAL FILE COUNT 3 FILE SIZE 60;
Systrk file TRACK01 page size 7,548 file size 60
delete NULL file count 3.
Systrk file TRACK02 page size 7,548 file size 60
delete NULL file count 3.
Systrk file TRACK03 page size 7,548 file size 60
delete NULL file count 3.
Status = 0  SQLSTATE = 00000
AutoCommit will COMMIT transaction
Command Facility ended with no errors or warnings
```

INSTALL STAMPS

The INSTALL STAMPS utility stores synchronization stamps in an area of an SQL-defined database that was reformatted by file.

This article describes the following information:

- Authorization (see page 106)
- Syntax (see page 107)
- Parameters (see page 107)
- Usage (see page 107)
- JCL Considerations (see page 108)
- Examples (see page 108)
- Sample Output (see page 108)

Authorization

To install stamps for an area, you need DBAWRITE access on the area.
Syntax

```plaintext
INSTALL STAMPS INTO AREA segment-name.area-name
SEGMENT segment-name
```

Parameters

- **AREA**
  Directs the INSTALL STAMPS utility to install synchronization stamps in an area.

- **segment-name**
  Specifies the name of the segment containing the area.

- **area-name**
  Specifies the name of the area.

- **SEGMENT**
  Specifies the segment whose areas will have their stamps installed.

- **segment-name**
  Specifies the name of a segment included in the DMCL module.

- **INITIAL**
  Specifies that the area(s) contain no synchronization stamps because they were formatted using the file or segment option of the FORMAT utility statement executing in local mode. INITIAL is the default.

- **REPLACE**
  Specifies that the area(s) contain synchronization stamps that should be replaced with those from the catalog.

Usage

**How to submit the INSTALL STAMPS statement**

You submit the INSTALL STAMPS statement using either the batch command facility or the online command facility.

**When to use INSTALL STAMPS**

Use the INSTALL STAMPS utility only if the area was reformatted using the file option of the FORMAT utility statement.

**When not to use INSTALL STAMPS**
If you have used the area or segment option of the FORMAT utility to initialize the area, do not run the INSTALL STAMPS utility. The synchronization stamps have already been installed.

Do not use the INSTALL STAMPS utility on areas of non-SQL-defined databases.

**Use caution when replacing stamps**

By replacing stamps in an area, you are asserting that the catalog’s definition accurately describes data in the area. You should be sure that this is true or that the area contains no data before replacing stamp values. No data validation is performed by the utility.

**JCL Considerations**

When you submit an INSTALL STAMPS statement through the batch command facility in local mode, the JCL to execute the facility must include statements to define the following:

- File(s) that map to the area to be processed
- Dictionary that defines the table(s) in the area
- Journal file(s) if backups are not taken

To run under central version, a SYSCTL statement is needed.

**Examples**

The following example directs the INSTALL STAMPS utility to store synchronization stamps in the EMPLDEMO.EMPLAREA area.

```
install stamps into area empldemo.emplarea;
```

**Sample Output**

After successfully storing synchronization stamps in the EMPLDEMO.EMPLAREA area, the following listing is generated.

```
IDMSBCF mm/dd/yy PAGE 1 IDMS Batch Command Facility

CONNECT TO SQLDEMO;
Status = 0
FORMAT FILE EMPLDEMO.EMPF1;
File EMPLDEMO.EMPF1 blocks 1 to 50.
Area EMPLDEMO.EMPLAREA pages 5,001 to 5,050.
Page size in file 4,096.
Status = 0
INSTALL STAMPS INTO AREA EMPLDEMO.EMPLAREA ;
Status = 0
AutoCommit will COMMIT transaction
Command Facility ended with no errors or warnings
```
LOAD

The LOAD utility loads data into an SQL-defined database.

The process of loading a database has three major phases:

<table>
<thead>
<tr>
<th>Phase</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load</td>
<td>Loads the data.</td>
</tr>
<tr>
<td>Build</td>
<td>Builds indexes and linked indexed referential constraints.</td>
</tr>
<tr>
<td>Validate</td>
<td>Ensures the validity of referential constraints.</td>
</tr>
</tbody>
</table>

Each phase is composed of multiple steps.

You can run the LOAD utility in several ways:

<table>
<thead>
<tr>
<th>Type of LOAD</th>
<th>What It Does</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete LOAD</td>
<td>Runs all three phases.</td>
</tr>
<tr>
<td>Phased LOAD</td>
<td>Runs either the LOAD and BUILD phases or just the LOAD phase.</td>
</tr>
<tr>
<td>Stepped LOAD</td>
<td>Runs the first or second step of the LOAD phase, with intermediate file sorting required between each step.</td>
</tr>
</tbody>
</table>

⚠️ **Note:** For more information about the phases and steps involved in loading a database, and help in deciding what kind of load you should run, see the *CA IDMS Database Administration Section*. 

This article describes the following information:

- Authorization (see page 110)
- Syntax (see page 110)
- Parameters (see page 111)
- Usage (see page 116)
- JCL Considerations (see page 117)
- Examples (see page 118)
- Sample Output (see page 119)
Authorization

<table>
<thead>
<tr>
<th>To</th>
<th>You need this privilege</th>
<th>On</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load into a table</td>
<td>INSERT</td>
<td>The table</td>
</tr>
</tbody>
</table>

⚠️ **Note:** Only one LOAD, BUILD, or VALIDATE statement can be performed during one execution of the Batch Command Facility (IDMSBCF).

Syntax

**Complete LOAD:**

**Syntax for complete or phased LOAD**

```plaintext
LOAD
  target-table-specification
  WITHOUT PRESORT
  NO BUILD
  BUILD INDEX ONLY
  NO VALIDATE
  FROM start-record
  FOR record-count
  LRECL logical-record-length
  NOTIFY notify-record-count
  INTO table-id
target-col-name POSITION start-pos data-type NULL IF literal
WHERE select-column-name = literal
  POSITION start-position
```

**Expanded syntax for target table specification**

```plaintext
schema-name.
```

```plaintext
ERRORS error-count CONTINUE
```

```plaintext
LOAD
  BUILD
  VALIDATE
  ALL
```
Stepped LOAD

Syntax for STEP 1

```
LOAD STEP1
  target-table-specification
  WITHOUT PRESORT
  EXTRACT INDEXES CONSTRAINTS BOTH
  FROM start-record
  FOR record-count
  LRECL logical-record-length
  NOTIFY notify-record-count
  ERRORS error-count CONTINUE
```

Syntax for STEP 2

```
LOAD STEP2
  NOTIFY notify-record-count
```

Parameters

Parameters for complete or phased LOAD

- **target-table-specification**
  Specifies the tables and columns to be loaded.
  See parameter descriptions under "Parameters for Target Table Specification".

- **WITHOUT PRESORT**
  Directs the LOAD utility not to sort the input data in SYS001 before starting to load.
  By default, if you do not specify WITHOUT PRESORT, the LOAD utility will sort the input data into target page sequence before beginning to load.

**Note:** For more information and help in deciding whether to suppress the presort, see the CA IDMS Database Administration Section.
NO BUILD
Directs the LOAD utility to perform neither the BUILD nor the VALIDATE phase.
If you specify NO BUILD, you must run the BUILD utility before you can use the table(s).

BUILD INDEX ONLY
Directs the LOAD utility to build indexes but not referential constraints.
If you specify BUILD INDEX ONLY, you might have to run the BUILD utility before you can use the table(s).

NO VALIDATE
Directs the LOAD utility to stop before validating referential constraints.
If you specify NO VALIDATE, you will have to run the VALIDATE utility before you can use the table(s).

FROM
Directs the LOAD utility to begin processing input data from a specified record in the input file.
By default, if you do not specify a start-record, the LOAD utility will begin with the first record of the input file.

start-record
Specifies the number of the first record to process.

FOR
Directs the LOAD utility to stop after processing a specified number of records from the input file.
By default, if you do not specify a FOR option, the LOAD utility will continue until it has processed the last input record.

record-count
Specifies the number of records to process before stopping.

ALL
Directs the LOAD utility to continue processing until it has loaded the last record in the input file.
All is the default.

LRECL
Specifies that the SYS001 input records are fixed length records.
By default, if you do not specify LRECL, the LOAD utility assumes that the SYS001 records are variable length.

logical-record-length
Specifies the length, in bytes, of the fixed length input records.

NOTIFY
Directs the LOAD utility to send a message to the operator whenever a specified number of records are processed.
The message states the phase and step currently being executed and the number of records processed.
By default, the LOAD utility will not notify you of its progress until it is finished.

notify-record-count
Specifies the number of records to process before sending a message.
ALL ERRORS
Directs the LOAD utility either to continue when any errors are detected or stop after a specified number of errors are detected. By default, the LOAD utility will stop after the first error is detected. Detected errors are listed in the report generated by LOAD and sent to the SYSLST file.

LOAD ERRORS
Directs the LOAD utility either to continue when errors are detected in the LOAD process or stop after a specified number of errors are detected. By default, the LOAD utility will stop after the first error is detected. Detected errors are listed in the report generated by the LOAD utility and sent to the SYSLST file.

BUILD ERRORS
Directs the LOAD utility either to continue when errors are detected while indexes and referential constraints are being built or stop after a specified number of errors are detected. By default, the LOAD utility will stop after the first error is detected. Detected errors are listed in the report generated by the LOAD utility and sent to the SYSLST file.

VALIDATE ERRORS
Directs the LOAD utility either to continue when errors are detected in the validation or stop after a specified number of errors are detected. By default, the LOAD utility will stop after the first error is detected. Detected errors are listed in the report generated by the LOAD utility and sent to the SYSLST file.

error-count
Specifies the number of errors to detect before stopping.

CONTINUE
Directs the LOAD utility to continue regardless of the number of errors detected.

Parameters for Target Table Specification

INTO
Specifies a table in which to load data.

schema-name
Specifies the name of the schema that contains the table.

table-id
Specifies the identifier of the table.

WHERE
Directs the LOAD utility to insert a row into the table from an input record only if a field beginning at a specified position in the input record, or the contents of an input column, equals a given literal value.

select-column-name
Specifies the name of the column whose contents must equal a literal value. The column name you specify must also be a column name of the table and must be named in the column list of the target table specification if target column names are used.
- **literal**
  - Specifies the value the column must contain.
  - The data type of the literal must be comparable to that of the column.

  **Note:** For more information about comparable data types, see the [CA IDMS SQL Reference Section](#).

- **POSITION start-pos**
  - Specifies the beginning position of the field whose value is to be tested.

- **literal**
  - Specifies the value the field must contain. The literal must be a character or hexadecimal literal value.

- **target-column-name**
  - Specifies the columns of the table for which input values are present in the input file.
  - If no column names are specified, the input file must contain values for all columns of the table and the order, data type, and null indications must exactly match those of the table as defined in the dictionary. VARCHAR and VARGRAPHIC input values must be preceded by a 2-byte binary length of the value.
  - If you specify more than one target-column-name, specify them in increasing order; the position of each must be greater than the sum of:
    - The position of the one listed before
    - `plus`
    - The size of the data type for the one listed before
      - The size of VARCHAR is the maximum length, plus 2.
      - The size of VARGRAPHIC is the maximum length times the number of bytes for one character, plus 2.

- **POSITION**
  - Specifies the position of the column value in the input record.

- **start-pos**
  - Specifies the position of the first byte of the value relative to one.

- **data-type**
  - Specifies the SQL data type of the column value in the input record.

  **Note:** For a list of SQL data types and for the SQL standards of data-type specification, see the [CA IDMS SQL Reference Section](#).
• **NULL IF**
  Directs the LOAD utility to substitute a null value if it encounters a specified input value.

• **literal**
  Specifies the value for which a null will be substituted. The data type of the literal must be comparable to the data type of the column.

• **SQLIND**
  Indicates that a null indicator immediately follows the data value on the input file.
  The possible values of the null indicator and their meanings are as follows:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary zeros</td>
<td>The column value is not null</td>
</tr>
<tr>
<td>Binary X'FF's</td>
<td>The column value is null</td>
</tr>
</tbody>
</table>

• **(length)**
  Specifies the length of the indicator. The length must be 1, 2, or 4.

**Parameters for STEP 1**

• **LOAD STEP1**
  Directs the LOAD utility to perform only the first step of the load process.

• **target-table-specification**
  Specifies the tables and columns to be loaded.
  For the syntax expansion and parameter descriptions of target-table-specification, see "Complete target-table-specification" earlier in this section.

• **WITHOUT PRESORT**
  Directs the LOAD utility not to sort the input data before starting to load.
  By default, if you do not specify WITHOUT PRESORT, the LOAD utility will sort the input data before beginning to load. The records are sorted so that target pages will be loaded in order.

  **Note:** For more information and help in deciding whether to suppress the presort, see the CA IDMS Database Administration Section.

If you specify WITHOUT PRESORT, then do not run LOAD STEP2. After LOAD STEP1 is completed, and you have sorted the output of LOAD STEP1, run the BUILD utility.

• **EXTRACT**
  Directs LOAD to extract information needed for building indexes or referential constraints.
  By default, if you do not specify otherwise, LOAD STEP1 will extract the information needed for building both indexes and indexed referential constraints. In this case, you can begin BUILD processing with BUILD STEP2.
INDEXES
Directs the LOAD utility to extract only the information needed for building indexes. Therefore, the LOAD utility will not extract the information needed for building indexed referential constraints.

CONSTRAINTS
Directs the LOAD utility to extract only the information needed for building linked indexed referential constraints. Therefore, the LOAD utility will not extract the information needed for building indexes.

BOTH
Directs the LOAD utility to extract the information needed for building both indexes and referential constraints. In this case, when you run the BUILD utility, you can begin with BUILD STEP2. BOTH is the default.

NO EXTRACT
Directs the LOAD utility not to extract information needed for building either indexes or referential constraints. In this case, run STEP1 of the BUILD utility to do the extraction.

Note: The remaining parameters are identical to the like-named parameters for a complete LOAD presented earlier in this section. Refer to this section for parameter descriptions.

Parameters for STEP 2

LOAD STEP2
Directs the LOAD utility to perform only the second step of the load process. Execute this only after executing LOAD STEP1 without specifying WITHOUT PRESORT.

Note: The remaining parameters are identical to the like-named parameters for a complete LOAD presented earlier in this section. Refer to this section for parameter descriptions.

Usage

How to submit the LOAD statement
You submit the LOAD statement only through the batch command facility. You must run the batch command facility in local mode.

How LOAD works
The LOAD utility reads records sequentially from an input file whose external name is SYS001.
The specifications in the syntax tell the LOAD utility how to interpret each record in SYS001 and which table(s) to load. The LRECL parameter controls whether the input file is fixed or variable length.

**When to use LOAD**

Use the LOAD utility to load an SQL-defined database for the first time or to expand it afterwards.

**When not to use LOAD**

Do not use the LOAD utility to load a non-SQL-defined database. Instead, use the FASTLOAD utility.

**When to use a phased or stepped LOAD**

Considerations for using the LOAD utility for a phased or stepped load and detailed loading procedures are discussed in the *CA IDMS Database Administration Section*.

**SYS001 input file contents**

The LOAD utility does not perform any type of data translation against the input records within the SYS001 file. As a result, each column value in the SYS001 file must be in the proper internal format as specified by the corresponding column's data format. For example, if a column has a format of DECIMAL(5), the corresponding field in the SYS001 record must be a valid packed decimal number occupying 3 bytes.

**Sorting intermediate work files**

If you run the load process in steps or phases, use the sort parameters in the SYSPCH file to sort the intermediate files.

**Checking error messages**

Use the *CA IDMS Messages and Codes Section* to locate messages associated with return codes received from the LOAD utility. Additionally, other useful information about any errors that occurred during LOAD processing are generated on the listing produced by the LOAD utility.

**JCL Considerations**

When you submit a LOAD statement to CA IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define:

- The dictionary containing table definitions
- The files containing the areas associated with the tables being loaded
- The intermediate work files
- Sort work files, if doing a complete LOAD

**LOAD utility uses intermediate work files**
Each step of the load process produces intermediate work files to be used by later steps. If you run a complete LOAD without separating steps or phases, the LOAD utility sorts data in the intermediate files between the steps automatically. If you run a phased or stepped LOAD, you must run the intermediate sorts.

**Note:** When running a complete or phased LOAD, SYS002 and SYS003 must point to the same intermediate file. When the database being processed is so large that the intermediate file must be a multi-volume data set, it is required that all extents be physically allocated before jobstep initiation. If this is not possible, then a stepped LOAD should be used. When running a stepped LOAD, SYS002 and SYS003 must point to different intermediate files. The data that is output in SYS003 by each step is input to the next step in SYS002.

The following table shows the output of LOAD STEP1 and LOAD STEP2:

<table>
<thead>
<tr>
<th>Step</th>
<th>Output</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEP1 (WITH PRESORT)</td>
<td>SYS003</td>
<td>For each record:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(MAX SORT CONTROL KEY SIZE) +</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(MAX FOREIGN KEY SIZE) +</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(MAX SCHEMA RECORD SIZE) + 24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYSPCH contains sort parameters</td>
</tr>
<tr>
<td></td>
<td></td>
<td>80 bytes</td>
</tr>
<tr>
<td>STEP1 (WITHOUT PRESORT)</td>
<td>SYS003</td>
<td>For each record:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(MAX SORT CONTROL KEY SIZE) +</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(MAX FOREIGN KEY SIZE) + 24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYSPCH contains sort parameters</td>
</tr>
<tr>
<td></td>
<td></td>
<td>80 bytes</td>
</tr>
<tr>
<td>STEP2</td>
<td>SYS003</td>
<td>For each record:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(MAX SORT CONTROL KEY SIZE) +</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(MAX FOREIGN KEY SIZE) + 24</td>
</tr>
</tbody>
</table>

**Note:** For more information about the generic JCL used to execute the batch command facility, see the section specific to your operating system.

### Examples

**Phased LOAD**

The following sample statement instructs the LOAD utility to load data from an input file into various tables when a record in the input file at position 60 matches the where clause criteria.

```
load into load.a where position 60 = 'a'
  into load.b where position 60 = 'b'
```
Complete LOAD

The next example instructs the LOAD utility to perform a complete load of data from an input file into various tables when a record in the input file at position 60 matches the where clause criteria. In addition, it will load data into two tables when a record in the input file matches the character string associated with the column name in the where clause.

```sql
load into load.c where position 60 = 'C'
    into load.d where position 60 = 'D'
    into load.e where position 60 = 'E'
    into load.f where position 60 = 'F'
    into load.g where position 60 = 'G'
    into load.m where mchar = 'm1'
        (mchar    position 1   char(2),
         mvchar   position 3   varchar,
         mbin     position 7   binary(2),
         mhalf    position 9   smallint,
         mfull    position 11  integer,
         mlong    position 15  longint,
         mdec     position 23  dec(2,1),
         mdecu    position 25  unsigned dec(3,3),
         mnum     position 27  num(2,1),
         mnumu    position 29  unsigned numeric(3,3),
         mdate    position 32  date,
         mtime    position 42  time,
         mts      position 50  timestamp)

    into load.m2 where position 1 = 'M2'
        (mgraph   position 3   graphic,
         mvgraph  position 5   vargraphic(4),
         mreal    position 15  real,
         mfloat1  position 19  float(4),
         mfloat2  position 23  float(50),
         mdp      position 31  double precision)
        errors continue
    lrecl 80;
```

Sample Output

Complete LOAD

The following report was generated after executing the LOAD statement in example 1 shown previously.

```
IDMSBCF IDMS Batch Command Facility

CONNECT TO SYSCAT;
Status = 0
SET BATCH
    HEADINGS OFF WIDTH PAGE 79 UNDERLINE '.'
    SOLCODE ERROR
    COMPRESS ON;
*DEBUG IDMS OFF

--    **** FORMAT segments    ****

FORMAT SEGMENT USERDB;
```

File USERDB.EMPF1 blocks 1 to 50.
Area USERDB.EMP_AREA pages 5,001 to 5,050.
Page size in file 4,096.
File USERDB.ORGF1            blocks 1 to 50.
Area USERDB.ORG_AREA         pages 5,051 to 5,100.
Page size in file 4,096.

File USERDB.EMPIX             blocks 1 to 50.
Area USERDB.EMPIX_AREA        pages 5,101 to 5,150.
Page size in file 2,000.

-- **** Load data into Tables ****

*DEBUG IDMS ON

LOAD INTO LOAD.A WHERE POSITION 60 = 'A'
INTO LOAD.B WHERE POSITION 60 = 'B'
INTO LOAD.H WHERE POSITION 60 = 'H'

NO VALIDATE LRECL 80;

IDMSLOAD - volser PRESORT TABLES FOR LOAD yy-mm-dd-hh.mm.ss
IDMSLOAD - 1 records processed for table LOAD.H
IDMSLOAD - 1 records processed for table LOAD.B
IDMSLOAD - 1 records processed for table LOAD.A
IDMSLOAD - 7 intermediate records were written to SYS003
IDMSLOAD - largest SYS003 record size is 136 characters
IDMSLOAD - PRESORT TABLES FOR LOAD processing completed
IDMSLOAD - volser LOAD TABLES AFTER SORT yy-mm-dd-hh.mm.ss
IDMSLOAD - 1 records processed for table LOAD.A
IDMSLOAD - 1 records processed for table LOAD.B
IDMSLOAD - 1 intermediate records for constraint LOAD.PKIX_B
IDMSLOAD - 1 intermediate records for constraint LOAD.FKIX_B
IDMSLOAD - 1 intermediate records for constraint LOAD.AB_S2
IDMSLOAD - 1 records processed for table LOAD.H
IDMSLOAD - 1 intermediate records for constraint LOAD.PKIX_H
IDMSLOAD - 3 records were stored in the database
IDMSLOAD - 12 intermediate records were written to SYS003
IDMSLOAD - largest SYS003 record size is 56 characters
IDMSLOAD - LOAD TABLES AFTER SORT processing completed
IDMSLOAD - volser FILL IN OWNERS DBKEY yy-mm-dd-hh.mm.ss
IDMSLOAD - 12 intermediate records were written to SYS003
IDMSLOAD - largest SYS003 record size is 56 characters
IDMSLOAD - FILL IN OWNERS DBKEY processing completed
IDMSLOAD - volser CONNECT UP INDEXES yy-mm-dd-hh.mm.ss
IDMSLOAD - 9 intermediate records were written to SYS003
IDMSLOAD - largest SYS003 record size is 56 characters
IDMSLOAD - CONNECT UP INDEXES processing completed
IDMSLOAD - volser BUILD RECORD PREFIX yy-mm-dd-hh.mm.ss
IDMSLOAD - BUILD RECORD PREFIX processing completed

AutoCommit will COMMIT transaction
Command Facility ended with warnings

Complete LOAD specifying column name

The next report was generated after executing the LOAD statement in example 2.

⚠️ **Note:** Some errors occurred during the load of tables LOAD.M and LOAD.M2. The LOAD utility provides useful information about the type and location of the error. Additional status information can be found in the *CA IDMS Messages and Codes Section*. 
*DEBUG IDMS OFF
CONNECT TO SYSCAT;
Status = 0
SET BATCH
    HEADINGS OFF WIDTH PAGE 79 UNDERLINE ' -'
    SOLCODE ERROR
    COMPRESS ON;

-- **** Load data into Tables ****
*DEBUG IDMS ON

LOAD INTO LOAD.C WHERE POSITION 60 = 'C'
INTO LOAD.D WHERE POSITION 60 = 'D'
INTO LOAD.E WHERE POSITION 60 = 'E'
INTO LOAD.F WHERE POSITION 60 = 'F'
INTO LOAD.G WHERE POSITION 60 = 'G'

INTO LOAD.M WHERE MCHAR = 'M1'
    (MCHAR    POSITION 1    CHAR(2),
     MVCHAR   POSITION 3    VARCHAR,
     MBIN     POSITION 7    BINARY(2),
     MHALF    Position 9    SMALLINT,
     MFULL    Position 11   INTEGER,
     MLONG    Position 15   LONGINT,
     MDEC     Position 23   DEC(3,3),
     MFECU    Position 25   UNSIGNED DEC(3,3),
     MNM     Position 27   NUM(2,1),
     MNMNU    Position 29   UNSIGNED NUMERIC(3,3),
     MDATE    Position 32   DATE,
     MTIME    Position 42   TIME,
     MTS      Position 50   TIMESTAMP)

INTO LOAD.M2 WHERE POSITION 1 = 'M2'
    (MGRAPH   Position 3    GRAPHIC,
     MVGRAPH  Position 5    VARGRAPHIC(4),
     MREAL    Position 15   REAL,
     MFLOAT1  Position 19   FLOAT(4),
     MFLOAT2  Position 23   FLOAT(50),
     MDP      Position 31   DOUBLE PRECISION)
ERRORS CONTINUE
LRECL 80;
IDMSLOAD - volser PRESORT TABLES FOR LOAD yy-mm-dd-hh.mm.ss
    SOLCODE = -4  Extended reason code = 1026  Messages follow:
DB001026 C.4M322: Data conversion error
IDMSLOAD - error in table LOAD.M - I/P record sequence number 10
IDMSLOAD - error in column 32 - column name is MDATE
    ERROR RECORD - +0 D4F10001 C2F1FFFF 00020000 00020000 M1..B1...........
    +10 E7F57667 E7F566E7 E7F57467 F3F046F0 XXX-XX-XXXX..30..0
    +20 F1F1F3F9 F060F0F1 60FF0F26 F1F64BF3 11390-01-02-16.3
    +30 F04BF0F1 4BF0F0FF F0F0FOFO 0.01.00000
SQLCODE = -4  Extended reason code = 1025  Messages follow:
DB001025 C.4M322: Data exception
IDMSLOAD - error in table LOAD.M2 - I/P record sequence number 16
IDMSLOAD - error in column 5 - column name is MVGRAPH
    ERROR RECORD - +0 D4F2C7F4 0004C7F1 0004C7C7 C7C70000 M2G4..G1G2GGGG.
    +10 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    +20 00000000 00000000
IDMSLOAD - 3 records processed for table LOAD.M2
IDMSLOAD - 3 records processed for table LOAD.M
IDMSLOAD - 1 records processed for table LOAD.M
IDMSLOAD - 1 records processed for table LOAD.G
IDMSLOAD - 1 records processed for table LOAD.F
IDMSLOAD - 1 records processed for table LOAD.E
IDMSLOAD - 1 records processed for table LOAD.D
IDMSLOAD - 1 records processed for table LOAD.C
IDMSLOAD - 19 intermediate records were written to SYS003
IDMSLOAD - largest SYS003 record size is 136 characters
IDMSLOAD - PRESORT TABLES FOR LOAD processing completed
IDMSLOAD - volser LOAD TABLES AFTER SORT yy-mm-dd-hh.mm.ss
IDMSLOAD - 1 records processed for table LOAD.C
IDMSLOAD - 1 intermediate records for constraint LOAD.FKIX_C
IDMSLOAD - 1 intermediate records for constraint LOAD.BC_S2
IDMSLOAD - 1 records processed for table LOAD.D
IDMSLOAD - 1 intermediate records for constraint LOAD.PKIX_D
IDMSLOAD - 1 intermediate records for constraint LOAD.FKIX_D
IDMSLOAD - 1 records processed for table LOAD.E
IDMSLOAD - 1 intermediate records for constraint LOAD.PKIX_E
IDMSLOAD - 1 records processed for table LOAD.F
IDMSLOAD - 1 intermediate records for constraint LOAD.FKIX_F
IDMSLOAD - 1 intermediate records for constraint LOAD.FKIX_FB
IDMSLOAD - 1 records processed for table LOAD.G
IDMSLOAD - 3 records processed for table LOAD.M
IDMSLOAD - 3 intermediate records for constraint LOAD.IX_M
IDMSLOAD - 3 records processed for table LOAD.M2
IDMSLOAD - 3 intermediate records for constraint LOAD.IX1_M2
IDMSLOAD - 3 intermediate records for constraint LOAD.IX2_M2
IDMSLOAD - 11 records were stored in the database
IDMSLOAD - 34 intermediate records were written to SYS003
IDMSLOAD - largest SYS003 record size is 56 characters
IDMSLOAD - LOAD TABLES AFTER SORT processing completed
IDMSLOAD - volser FILL IN OWNERS DBKEY yy-mm-dd-hh.mm.ss
IDMSLOAD - 34 intermediate records were written to SYS003
IDMSLOAD - largest SYS003 record size is 56 characters
IDMSLOAD - FILL IN OWNERS DBKEY processing completed
IDMSLOAD - volser CONNECT UP INDEXES yy-mm-dd-hh.mm.ss
IDMSLOAD - 19 intermediate records were written to SYS003
IDMSLOAD - largest SYS003 record size is 56 characters
IDMSLOAD - CONNECT UP INDEXES processing completed
IDMSLOAD - volser BUILD RECORD PREFIX yy-mm-dd-hh.mm.ss
IDMSLOAD - BUILD RECORD PREFIX processing completed
IDMSLOAD - volser VALIDATE INDEXES STEP 1 yy-mm-dd-hh.mm.ss
IDMSLOAD - 1 records processed for table LOAD.C
IDMSLOAD - 0 intermediate records for constraint LOAD.BC_S1
IDMSLOAD - 0 intermediate records for constraint LOAD.BC_S2
IDMSLOAD - 0 records processed for table LOAD.D
IDMSLOAD - 0 intermediate records for constraint LOAD.LOAD_DD
IDMSLOAD - 1 intermediate records for constraint LOAD.LOAD_AD
IDMSLOAD - 2 records processed for table LOAD.E
IDMSLOAD - 0 intermediate records for constraint LOAD.LOAD_AE
IDMSLOAD - 0 records processed for table LOAD.F
IDMSLOAD - 0 intermediate records for constraint LOAD.LOAD_EF
IDMSLOAD - 0 intermediate records for constraint LOAD.LOAD_BF
IDMSLOAD - 1 intermediate records for constraint LOAD.LOAD_HF
IDMSLOAD - 0 intermediate records for constraint LOAD.LOAD_BG
IDMSLOAD - 0 records processed for table LOAD.M
IDMSLOAD - 0 records processed for table LOAD.M2
IDMSLOAD - 19 intermediate records were written to SYS003
IDMSLOAD - largest SYS003 record size is 56 characters
IDMSLOAD - VALIDATE INDEXES STEP 1 processing completed
IDMSLOAD - volser VALIDATE INDEXES STEP 2 yy-mm-dd-hh.mm.ss
IDMSLOAD - VALIDATE INDEXES STEP 2 processing completed

Status = 1 Extended Reason Code = 2991 Messages follow:
DB002991 C1M349: Error detected doing a LOAD/BUILD/VALIDATE statement

AutoCommit will COMMIT transaction

Command Facility ended with warnings

More Information

- For more information about procedures for loading a database, see the *CA IDMS Database Administration Section*. 
For more information about SQL data-type specifications, see the CA IDMS SQL Reference Section.

**LOCK**

The LOCK utility allows a DBA to explicitly lock an area. This allows the DBA to place a lock on an area that will remain in effect across several commands. Access to an area can be prevented while a series of operations is performed on it.

This article describes the following information:

- Authorization (see page 123)
- Syntax (see page 123)
- Parameters (see page 123)
- Usage (see page 124)
- JCL Considerations (see page 124)
- Examples (see page 124)
- Sample Output (see page 125)

**Authorization**

To lock an area, you need DBAWRITE access on the area.

**Syntax**

```
LOCK AREA segment-name.area-name SEGMENT segment-name EXCLUSIVE UPDATE
```

**Parameters**

- **AREA**
  Directs the LOCK utility statement to lock a specified area.

- **segment-name**
  Identifies the name of the segment associated with the area to be locked.

- **area-name**
  Identifies the name of the area to be locked.

- **SEGMENT**
  Specifies the segment whose areas will be locked.

- **segment-name**
  Identifies the name of the segment to be locked.
EXCLUSIVE UPDATE

Specifies the update mode. EXCLUSIVE UPDATE is the default mode and the only mode currently supported.

Usage

Local mode execution

If the LOCK AREA statement is issued through the batch command facility executing in local mode, a physical lock will be placed on the area. The lock remains in effect until an explicit UNLOCK is issued. If the area is already locked, the LOCK statement will fail with a DB002352 error message as illustrated in the sample output.

If the LOCK SEGMENT statement is issued in local mode, a physical lock will be placed on each area in the segment that is not already locked. A status of each area will be reported. Once all areas have been locked or checked, if any area in the segment was already locked, a DB002406 error message will be issued, as illustrated in the sample output.

Online execution

If the LOCK statement is issued through the online command facility (OCF) or through the batch command facility executing in a batch/CV mode, a logical lock will be placed on the area. This lock prevents all access to the area by other users until the session in which the LOCK statement is issued is committed. If executing online, a commit is automatically issued at end of task prior to the pseudo-converse unless autocommit is disabled through a SET OPTIONS statement.

The areas being locked must be varied in update mode to the central version. If another task is updating an area being locked, the LOCK command will wait until the logical area lock is available. If the wait would cause a deadlock, the LOCK command fails.

Committing prior work

Before executing this utility under a central version, you must commit any previous work done within the current session. For more information, see Central Version Considerations (see page 24).

JCL Considerations

When you submit a LOCK statement through the batch command facility in local mode, the JCL to execute the facility must include statements to define the file containing the first page of the area to be processed. To run under central version, a SYSCTL statement is needed.

Examples

Lock by area

The following example directs the LOCK utility to lock the EMPDEMO.EMP-DEMO-REGION area.

`lock area empdemo.emp-demo-region;`
Lock by segment

The following example directs the LOCK utility to lock the EMPDEMO segment.

lock segment empdemo;

Sample Output

Lock by area

When LOCK processing is completed, the following listing is generated when processing is successful.

IDMSBCF nn.n CA IDMS Batch Command Facility mm
/dd/yy PAGE 1
LOCK AREA EMPDEMO.EMP-DEMO-REGION;
Status = 0 SQLSTATE = 00000

When LOCK processing is completed, the following listing is generated when processing fails.

IDMSBCF nn.n CA IDMS Batch Command Facility mm
/dd/yy PAGE 1
LOCK AREA USERDB.EMP_AREA;
Status = -4 SQLSTATE = 5000B Messages follow:
DB002352 C-4M353: Area USERDB.EMP_AREA required area lock mode not available

Lock by segment

When LOCK processing is completed, the following listing is generated.

IDMSBCF nn.n CA IDMS Batch Command Facility mm
/dd/yy PAGE 1
LOCK SEGMENT EMPDEMO;
Area EMPDEMO.EMP-DEMO-REGION was not available
Area EMPDEMO.INS-DEMO-REGION Lock Mode Acquired
Area EMPDEMO.ORG-DEMO-REGION Lock Mode Acquired
Status = -4 SQLSTATE = 28000 Messages follow:
DB002406 C-4M363: Area lock mode unavailable for one or more areas

MAINTAIN INDEX

The MAINTAIN INDEX utility builds, rebuilds, or deletes one or more indexes in a non-SQL-defined database.

You can run the MAINTAIN INDEX utility all at once or break it into steps.

This article describes the following information:

- Authorization (see page 126)
- Syntax (see page 126)
- Parameters (see page 126)
- Usage (see page 129)
Authorization

To process an index in a segment, you need the privilege DBAWRITE, on all areas of the segment.

Syntax

```
►►─── MAINTAIN INDEX ──────────────────────────────────────────────────────────►
►─── IN ─┬─ SEGMENT segment-name ───────────────────────────►
           └─ DBNAME database-name ──────────────────────────►
►─── USING subschema-name ────────────────────────────────►
           └─ NEWSUB new-subschema-name ───────────────────►
►─── NOTIFY notify-record-count ───────────────────────────►
           └─ STEP ─┬─ step-name ──►
              └─ FROM ─┘
         ┌─────────────────────────────────────────────┐
►─▼─┬─ BUILD ──┬─ INDEX ──┬─ REBUILD ──►
     │      └─ index-name ─┬─ index-name ─┐
     │                   └─ INDEX ──┐
     └─ INDEX ─┬─ ALLrows ──►
                └─ MEMbers ─┘
               ┌──  as SORTEXIT ─┐
               └──  REUSE workfiles ┘
```

Parameters

- **IN SEGMENT**
  Specifies the segment containing the index(es) to be processed.

- **segment-name**
  Specifies the name of the segment.

- **IN DBNAME**
  Specifies the database containing the index(es) to be processed.

- **database-name**
  Specifies the name of the database.

- **USING**
  Specifies the subschema that defines the index(es).
Note: If processing an index associated with an ASF table, specify the name of the table’s default subschema, RUNnnnnn where nnnnnn is the table’s definition number preceded by zeros.

If processing an ASF index that resides in ASF’s definition area, IDMSR-AREA, which specify the IDMSRSSA subschema.

- **subschema-name**
  Specifies the name of the subschema.
  - If building an index, this is the name of the subschema defining the index to be built.
  - If deleting an index, this is the name of the subschema defining the index to be deleted.
  - If rebuilding an index, this is the name of a subschema describing either the existing index structure or the new index structure.

- **NEWSUB**
  Specifies the subschema that defines one or more indexes to be rebuilt. Use this clause only when rebuilding an index according to a new definition.

- **new-subschema-name**
  Specifies the name of the new subschema.

- **NOTIFY**
  A message is sent to the system console after a specified number of records have been processed in the current step.

- **notify-record-count**
  Specifies the number of records to process before sending a message.
  By default or if 0 is specified, no message is sent to the system console except the standard message sent at the end of each step indicating the number of records processed during the step. Notify messages are displayed by steps IDMSTABX, IDMSDBL3, and IDMSDBL4 when a notify-record-count is specified.

- **STEP**
  Directs the MAINTAIN INDEX utility to execute only one step of the index maintenance process. By default, if you do not specify STEP, all steps are performed.
  If the specified step is any step except IDMSTABX, all other parameters are ignored. In this case, the information normally provided by the other parameters is obtained from the intermediate work files.

- **FROM**
  Specifies that MAINTAIN INDEX processing should begin at a specified step and complete all remaining steps.
  If the specified restart step is any step except IDMSTABX, all other parameters are ignored. In this case, the information normally provided by the other parameters is obtained from the intermediate work files.

- **step-name**
  Specifies the name of the first or only step to execute.
  The name must be one of the following:
* IDMSTABX
* SORT3
* IDMSDBL3
* SORT4
* IDMSDBL4

**BUILD**
For system-owned indexes only, directs the MAINTAIN INDEX utility to add all member record occurrences to the specified index.
Both the index and all associated areas and files must be defined in the subschema specified in the USING SUBSCHEMA-NAME clause.

**DELETE**
For system-owned indexes only, directs the MAINTAIN INDEX utility to:
- Disconnect all members from the specified index
- Remove the SR7 and SR8 records for the index from the database

**REBUILD**
For system-owned indexes only, directs the MAINTAIN INDEX utility to rebuild an existing, non-empty index.

**index-name**
Specifies the name of the index to process.

**FROM**
Identifies the records to use in rebuilding an index.
Use this parameter only with REBUILD.

**INDEX**
Directs the MAINTAIN INDEX utility to connect only members of the existing index to the new index.

**ALLrows**
Directs the MAINTAIN INDEX utility to:
- Sweep the area of the database that contains eligible members
- Connect all eligible member occurrences to the rebuilt index

**MEMbers**
Directs the MAINTAIN INDEX utility to:
- Sweep the area of the database that contains the member record type
- Determine if record occurrences found, participate in the index and if they do, connect them to the rebuilt index
- Connect all member record occurrences to the rebuilt index (this is the same as the ALLROWS option) for unlinked system-owned indexes only

- as SORTEXIT
  Causes each DBLx step in the utility to return its input data directly from the preceding sort instead of having the sort write the data to a workfile. This option eliminates one workfile for each sort and saves the I/O it takes to write, then read, the workfile.

- REUSE workfiles
  Causes each step in the utility to reuse an existing workfile, if possible, when writing its output data, instead of writing to a new one for each step. This reduces the number of workfiles that need to be allocated.

Usage

How to submit the MAINTAIN INDEX utility

You submit the MAINTAIN INDEX utility only through the batch command facility. You must run the batch command facility in local mode.

All areas affected by the index must be varied offline.

When to use MAINTAIN INDEX

Use the MAINTAIN INDEX utility to process indexes in a non-SQL-defined database. This includes indexes associated with individual ASF tables and ASF indexes that reside in the IDMSR-AREA; specify the IDMSRSSA subschema.

When not to use MAINTAIN INDEX

To process indexes in an SQL-defined database, use the LOAD or BUILD utility.

Multiple operations in one execution

You can perform maintenance on multiple indexes in one execution of the MAINTAIN INDEX utility. However, you should perform only one operation on an index within the same execution. For example, do not DELETE and BUILD the same index at the same time.

Subschema reentrancy

If the subschemas specified in the MAINTAIN INDEX utility reside in a load library, they must not be linked with the reentrant attribute, nor can they reside in the LPA (OS) or SVA (DOS). If the subschemas are loaded from a dictionary load area, these issues are not relevant.

BUILD-option with non-MA indexes

Specifying the BUILD-option will connect every potential member record to the system-owned index. Therefore, if the index is not MANDATORY AUTOMATIC and there are occurrences of the member record that should not participate in the index, do not use the BUILD-option. Instead, use the REBUILD FROM MEMBERS-option.

Rebuilding indexes
When rebuilding an index, MAINTAIN INDEX might need a subschema that describes the old index, the new index, or both depending on the changes (if any) being made to the index structure. The following table specifies the REBUILD option and subschema to use based on the function that the rebuild operation is performing:

<table>
<thead>
<tr>
<th>Function</th>
<th>REBUILD Option</th>
<th>Subschema Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reorganize an existing index (for example, after deleting many member occurrences)</td>
<td>FROM INDEX</td>
<td>USING old-subschema-name</td>
</tr>
<tr>
<td>Rebuild a damaged index</td>
<td>FROM MEMBERS</td>
<td>USING old-subschema-name</td>
</tr>
<tr>
<td>Modify the following index tuning options:</td>
<td>FROM INDEX</td>
<td>USING old-subschema-name</td>
</tr>
<tr>
<td>- Key compression</td>
<td></td>
<td>NEW SUB new-subschema-name</td>
</tr>
<tr>
<td>- Number of entries in an SR8 record</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Index displacement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Location of an index (its area or page range)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Linked or unlinked attribute</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Change the following index characteristics:</td>
<td>FROM MEMBERS</td>
<td>USING new-subschema-name</td>
</tr>
<tr>
<td>- Sort key</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Collating sequence</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Sorted or unsorted attribute</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Duplicates option</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rebuild an index from all member occurrences without making index changes</td>
<td>FROM ALLROWS</td>
<td>USING old-subschema-name</td>
</tr>
<tr>
<td>Rebuild an index from all member occurrences making index changes</td>
<td>FROM ALLROWS</td>
<td>USING new-subschema-name</td>
</tr>
</tbody>
</table>

Where there is a choice between using FROM INDEX or FROM MEMBERS, consider the following:

- FROM INDEX is generally more efficient because only the index structure is read, rather than every member record occurrence
- FROM INDEX preserves the order of entries with duplicate key values; FROM MEMBERS or FROM ALLROWS rebuilds the index with duplicate entries in db-key sequence
- FROM INDEX requires that the index exist and be readable

**Changing symbolic index values**

The following values can be supplied as a symbolic index parameter in the physical area definition:

- The number of entries in an SR8 (INDEX BLOCK CONTAINS)
- The number of pages bottom-level SR8s are displaced from top-level SR8s (DISPLACEMENT)

If these values are changed in the physical definition, you can (if desired) reorganize the index to reflect the new values by rebuilding it using a DMCL containing the updated segment definition.
Changing subarea page range

The page range in which a system-owned index resides can be specified by using a symbolic subarea parameter in the physical area definition. If this value is changed, execute MAINTAIN INDEX twice:

- The first execution must use a DMCL with the old values for the symbolic parameters and specify STEP IDMSTABX to indicate that only the first step of index maintenance is to be performed.

- The second execution must use a new DMCL (with the same name as the old DMCL) and specify FROM SORT3 to indicate that all remaining steps of index maintenance are to be performed.

To ensure that the correct DMCL is used in each case, you can either change load libraries or rename the new DMCL to have the same name as the old DMCL.

Adding or removing indexes

If adding or removing a linked system-owned index or a user-owned index set, you must use RESTRUCTURE to add or remove index and optional owner pointers in the member record.

Sorted indexes

Adding, removing, or changing the sort key of a sorted index (system- or user-owned) may change the control length of the record. If it does, and the record is compressed or variable in length, you must also use RESTRUCTURE to adjust the control length of the record in the database.

Duplicate index entries

When building an index (or rebuilding an index with the FROM ALLROWS or MEMBERS option), MAINTAIN INDEX stores duplicate index entries in the order in which the corresponding member record occurrences exist in the database (that is, in db-key sequence). When rebuilding an index with the FROM INDEX option, the order of duplicate index entries is maintained in the rebuilt index.

SORTEXIT and FROM/STEP

When using the FROM and STEP options with the SORTEXIT option, each pair of SORTn and DBLx steps are considered to be one step. If either half of the SORTn/DBLx is specified on a FROM or STEP option, processing will start with the SORTn step and the DBLx step will also be executed. For example:

- FROM IDMSDBL3 will start with step SORT3 and will continue to the end.

- STEP SORT3 will run steps SORT3 and IDMSDBL3.

SORTEXIT/REUSE WORKFILE restart considerations

Since SORTEXIT combines each SORTn step with the DBLx step that follows it, if a failure occurs in the DBLx step, a restart (if a restart is possible) must begin with the sort step and the input to the step will be resorted. Non-SORTEXIT mode will take longer to run but can be restarted after the sort in this case. Therefore, if restart time is more critical than normal runtime do not run the utility as a sortexit.

If the REUSE WORKFILE option is used with SORTEXIT, some input workfiles will be used as output files in the same step. Therefore if these two options are used together and a failure occurs, the utility must be restarted from the beginning.
Workfile Considerations for restarting a failed MAINTAIN INDEX

If the MAINTAIN INDEX command fails, depending on the reason for failure, restart the command at the failing step using the "FROM step-name" syntax. You can only restart a step if the input files to that step are intact and valid.

To prepare for a possible restart when running a one-step MAINTAIN INDEX, the Intermediate work files should have a disposition that preserves the data set in the event of an abend, for example, "DISP=(NEW,CATLG,CATLG)."

To restart MAINTAIN INDEX at a particular step, the input files to that step must have a disposition to specify that the files already exist, for example, "DISP=OLD".

To determine which files were input to a given step, see the "Intermediate Work File" tables under "JCL Considerations". Partially created output files should be deleted before restarting the job, and the original disposition should be used in the restart job, for example, "DISP=(NEW,CATLG,CATLG)".

The SYSPCH file contains sort parm information for sort steps. It is an output file to IDMSDBLn steps but is not read unless restarting or running in step mode. So during a normal run the SYSPCH file should be treated as a normal output file, for example, "DISP=(NEW,CATLG,CATLG)." However, restarting is not as straightforward. If the previous job failed in an IDMSDBLx step, the SYSPCH file was an output file and should be deleted before restarting. But if the failure occurred in a SORTx step, the contents of the SYSPCH file should contain the same values that were input to the SORTx step. In this case the SYSPCH file should be preserved and defined as a SYS001 input file to the restart step.

When the SORTEXIT option is used, the SORTx and IDMSDBLx steps are combined. If a failure occurs in this mode, the SYSPCH file should normally be preserved and used as a SYS001 input file to the restart. However, there is a small window at the end of a IDMSDBLx step where the SYSPCH file is opened for output and new SORT parameters are written. If the job fails at this point, the entire SORT x/IDMSDBLx step must be restarted, but the SYSPCH file will not be valid as a SYS001 input file. In this case, the sort parameters must be recreated by hand or the job must be restarted at an earlier IDMSDBLx step if possible. One way to avoid this situation is to run in step mode when running SORTEXIT mode.

The RELDCTL data set is always an input file to the first step of a MAINTAIN INDEX whether being restarted or not.

The steps of MAINTAIN INDEX

The MAINTAIN INDEX utility consists of the following steps, which you can run separately or as a single operation. Each step has an input and output:

<table>
<thead>
<tr>
<th>Step Description</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDM STABX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interprets the control statements</td>
<td>Control parameters from your submission of the utility</td>
<td>SYS003 contains index membership information</td>
</tr>
<tr>
<td>Reads the database</td>
<td>The database</td>
<td>SYSPCH contains sort parameters</td>
</tr>
<tr>
<td>Deletes system-owned indexes</td>
<td></td>
<td>RELDCTL file</td>
</tr>
<tr>
<td>Writes index descriptors to the SYS003 file</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Step</td>
<td>Description</td>
<td>Input</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOR</td>
<td>Sorts the contents of the SYS003 file and puts the results in SYS004.</td>
<td>SYS003 from IDMSTABX&lt;br&gt;SYS001 contains the sort parameters from IDMSTABXSYSPCH file.</td>
</tr>
<tr>
<td>T3</td>
<td></td>
<td>If running in step mode or if it is the first step in a sort, no sort parameters are needed.</td>
</tr>
<tr>
<td>IDM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOR</td>
<td>Sorts the contents of the SYS005 file, and puts the results in SYS006.</td>
<td>SYS005 from IDMSTABL3&lt;br&gt;SYS001 contains the sort parameters from IDMSTABL3SYSPCH file.</td>
</tr>
<tr>
<td>T4</td>
<td></td>
<td>If running in step mode or if this is the first step in a sort, sort control parameters are not needed.</td>
</tr>
<tr>
<td>IDM</td>
<td>Fills in prefix pointers for user-owned indexes and linked system-owned indexes.</td>
<td>SYS006 from SORT4&lt;br&gt;RELDCTL file</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The above table describes the input and output files when you execute MAINTAIN INDEX without the SORTEXIT and REUSE options. For the impact of running MAINTAIN INDEX with these options, see "JCL Considerations" later in this section.

Sort output after each step
If you execute the MAINTAIN INDEX utility a step at a time, you must use the sort parameters to sort the contents of the intermediate work files. You can use your own sort program or IDMSSORT.

Sort the intermediate work files as follows:

<table>
<thead>
<tr>
<th>Sort name</th>
<th>File to sort</th>
<th>Sort order</th>
<th>Sort on</th>
<th>Begins at</th>
</tr>
</thead>
<tbody>
<tr>
<td>SORT3</td>
<td>SYS003</td>
<td>Ascending</td>
<td>16 + (2 x n)</td>
<td>Byte 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>n is the length of the longest sort or CALC key in the subschema</td>
<td></td>
</tr>
<tr>
<td>SORT4</td>
<td>SYS005</td>
<td>Ascending</td>
<td>16 bytes</td>
<td>Byte 5</td>
</tr>
</tbody>
</table>

Note: If running in step mode, the sort parameters generated by IDMSTABX and IDMSDBL3 are not sufficient for stand-alone sort programs running under z/VSE. If you want to use your own sort program, you must add "WORK=" parameters to specify more than one sort work file.

Maintaining indexes

General procedure

In general, the procedure for changing indexes is as follows:

**Step 1: Create a new schema and global subschema, if necessary**

Note: These steps are not necessary if the only change being made is to the value of symbolic parameters associated with the index.

1. Create a new schema that is identical to the original schema.
2. Create a global subschema for the new schema with a name that is different from that of any other subschema in the dictionary. Include in the subschema all areas, records, and sets associated with the schema.
3. Make the necessary changes to the new schema definition.
4. Validate the schema.
5. Regenerate the global subschema.

**Step 2: Modify the segment and DMCL if necessary**
Note: Segment and DMCL modification is necessary only if adding or changing the values of symbolic index or subarea parameters associated with the index or adding a new area in which to store the index.

1. Make the appropriate changes in the segment definition. Make sure that subareas and other symbolics are defined appropriately.

2. Generate, punch, and link all DMCLs containing the altered segment.

Step 3: Make changes to the index

1. Backup the area(s).

2. Use the MAINTAIN INDEX utility to change a system-owned index.

3. Use a user-written program in conjunction with IDMSTBLU and the MAINTAIN INDEX utility to change a user-owned index.

4. Verify the change with IDMSDBAN or a retrieval program, CA OLQ or CA Culprit.

5. Back up the altered area(s).

Step 4: Complete the change

If schema changes were necessary:

1. Update the original schema in the same way that the copy was changed.

2. Regenerate all subschemas associated with the original schema that are affected by the change.

3. Recompile all access modules affected by the change, using the ALTER ACCESS MODULE statement with the REPLACE ALL option.

As appropriate, make the new subschemas, DMCL(s), and area(s) available to your runtime environment.

Maintaining user-owned indexes

To build, rebuild, or delete user-owned indexes, you must write a program which calls IDMSTBLU and passes information about the indexes to be operated on and the owner and member record occurrences participating in the indexes.

Once the program executes, complete the operation by executing the MAINTAIN INDEX utility, specifying FROM SORT3. Use the SYSPCH, SYS002, and RELDCTL work files generated by IDMSTBLU as input to the MAINTAIN INDEX utility as SYS001, SYS003, and RELDCTL respectively.

Creating the user-written program

The following considerations apply when writing your program:

1. Include in your program the following descriptors:
The global subschema describing the index

- The owner and member record descriptions
- Descriptions of the IDMSTBLU parameters outlined next

2. The general logic of the program should:

- As the first call to IDMSTBLU, pass the subschema descriptor:
  
  CALL IDMSTBLU USING SUBSCTYP.

- For each occurrence of a user-owned indexed set, identify the owner by passing an owner descriptor:
  
  CALL IDMSTBLU USING OWNERTYP.

Note: If more than one set per owner is to be processed during a single execution of the user-written program, multiple owner descriptors must be passed to IDMSTBLU. For example, to rebuild two indexed sets in which REC-A is owner and REC-B is member, as the second and third calls to IDMSTBLU, for example:

CALL IDMSTBLU USING OWNERTYP.
  
  ............+2 REBUILD IXSET-1 REC-A's dbkey
CALL IDMSTBLU USING OWNERTYP.
  
  ............+2 REBUILD IXSET-2 REC-A's dbkey

In this example, subsequent REC-B member descriptors passed to IDMSTBLU should contain two occurrences of set name and owner dbkey information (one occurrence for IXSET-1 and one occurrence for IXSET-2).

- For each record that participates as a member of an indexed set to be processed, pass a member descriptor and the member record occurrence:
  
  CALL IDMSTBLU USING MEMBERTYP member-record.

- As the last call to IDMSTBLU, pass the end-of-file descriptor:
  
  CALL IDMSTBLU USING EOFTYP.

The owner and member information can be obtained either from the database or a user input file.

   As information is passed to IDMSTBLU, it creates work files needed to build, rebuild, or drop the specified indexed sets.

3. Link edit your program with IDMSTBLU. Also, link it with IDMS if your program binds a rununit.

IDMSTBLU parameters

Subschema descriptor

The subschema descriptor identifies the subschema and segment that contain the indexed sets. If you are rebuilding an index and changing its characteristics, the identified subschema and segment must describe the new index definition.
The subschema descriptor also identifies the name of the DMCL to be used during MAINTAIN INDEX execution. It must be the same as the name of the DMCL specified in the SYSIDMS parameter file used to execute the MAINTAIN INDEX utility.

<table>
<thead>
<tr>
<th>Descriptor type</th>
<th>Fullword binary value 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subschema name</td>
<td>8-byte character</td>
</tr>
<tr>
<td>Segment name</td>
<td>8-byte character</td>
</tr>
<tr>
<td>DMCL name</td>
<td>8-byte character</td>
</tr>
</tbody>
</table>

For example:

```plaintext
01 SUBSCTYP.
  02 FILLER PIC S9(8) COMP VALUE +1.
  02 ssnm PIC X(8) VALUE 'EMPSS01'.
  02 segnm PIC X(8) VALUE 'EMPSEG'.
  02 dmclnm PIC X(8) VALUE 'NEWDMCL'.
```

**Owner descriptor**

The owner descriptor describes each record that participates as an owner in an indexed set to be processed:

<table>
<thead>
<tr>
<th>Descriptor type</th>
<th>Fullword binary value 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requested function</td>
<td>8-character value:</td>
</tr>
<tr>
<td></td>
<td>'BUILD', 'REBUILD', 'DELETE', 'EXTEND'</td>
</tr>
<tr>
<td>Set name</td>
<td>16-character set name</td>
</tr>
<tr>
<td>Owner db-key</td>
<td>Fullword binary owner db-key (zero if owner is an SR7 record)</td>
</tr>
</tbody>
</table>

For example:

```plaintext
01 OWNERTYP.
  02 FILLER PIC S9(8) COMP VALUE +2.
  02 OFUNC PIC X(8) VALUE SPACES.
  02 OSET PIC X(16) VALUE SPACES.
  02 ODBKEY PIC S9(8) COMP SYNC VALUE +0.
```

**Member descriptor**

The member descriptor describes each record that participates as a member in one or more indexed sets to be processed:

<table>
<thead>
<tr>
<th>Descriptor type</th>
<th>Fullword binary value 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record name</td>
<td>16-character record name</td>
</tr>
<tr>
<td>Record db-key</td>
<td>Fullword binary db-key</td>
</tr>
<tr>
<td>Number of sets</td>
<td>Fullword binary value that specifies the number of sets in which the record participates as a member</td>
</tr>
</tbody>
</table>
Repeat the following two fields once for every indexed set in which the record participates as a member:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set name</td>
<td>16-character set name</td>
</tr>
<tr>
<td>Owner db-key</td>
<td>Fullword binary owner db-key value</td>
</tr>
</tbody>
</table>

For example, the following illustrates a member that participates in user-owned indexed sets:

```plaintext
01 MEMBRTP.
  02 FILLER PIC S9(8) COMP VALUE +3.
  02 MREC PIC X(16) VALUE SPACES.
  02 MDBKEY PIC S9(8) COMP SYNC VALUE +0.
  02 MSETS PIC S9(8) COMP VALUE +2.
  02 MSET-INFO OCCURS n TIMES.
    04 MSET PIC X(16).
    04 MODBKEY PIC S9(8) COMP SYNC.
```

**Note:** n represents the number of indexed sets that are actually being processed (for example, those for which an owner descriptor was previously passed to IDMSTBLU and in which MREC participates as a member).

**Database record descriptor**

The database record descriptor describes each member record type in an indexed set to be processed:

For example:

```plaintext
01 COPY IDMS RECORD EMPLOYEE.
01 COPY IDMS RECORD SKILL.
01 COPY IDMS RECORD EXPERTISE.
01 COPY IDMS RECORD JOB.
```

**End-of-file descriptor**

The end-of-file descriptor serves as an end-of-file indicator:

<table>
<thead>
<tr>
<th>Descriptor type</th>
<th>Fullword binary value -1 (X'FFFFFFFF')</th>
</tr>
</thead>
</table>

For example:

```plaintext
01 EOFTYP.
  02 FILLER PIC S9(8) COMP VALUE -1.
```

**JCL Considerations**

When you submit a MAINTAIN INDEX utility through the batch command facility, the JCL to execute the facility must include statements to define:

- The files containing the areas to be processed
The intermediate work files

Sort space

**Note:** For more information about the generic JCL used to execute the batch command facility, see the section for your operating system in this section.

**Work file JCL considerations for STEP mode**

MAINTAIN INDEX normally runs as a single step but runs as separate steps using the "STEP step-name" syntax. When running in step mode, input files should have dispositions that state the file already exists, for example, "DISP=OLD"

Preserve output files on successful completion but not when the job fails, for example, "DISP=(NEW, CATLG,DELETE)".

See the "Intermediate Work File" table to determine which files are input and which files are output and when they are used.

The RELDCTL file is always input to every step.

The SYSPCH file is created by an IDMSDBLx step and used as input to a SORTx step. When used as input, it is defined as SYS001.

**Work file record lengths:**

- The RELDCTL file is a fixed length file with a record length of 60 bytes.
- The SYSPCH file is a fixed length file with a record length of 80 bytes.
- All SYSxxx files are variable length files. The record length can vary from one step to the next, from one job to the next. Do not code an LRECL value in the JCL just code a BLKSIZE value. A BLKSIZE value should be chosen based on the optimal size for the device being used, for example, 1/2 track if disk or 32k if tape.

**Intermediate work files**

The following tables indicate which work files are created and read by the different utility steps depending on the use of the SORTEXIT and REUSE WORKFILE options.

<table>
<thead>
<tr>
<th>Step</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAINTAIN INDEX: NOT sortexit mode and NOT reusing workfiles</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IDMSTABX</td>
<td></td>
<td>SYS003</td>
</tr>
<tr>
<td>SORT3</td>
<td>SYS003</td>
<td>SYS004</td>
</tr>
<tr>
<td>IDMSDBL3</td>
<td>SYS004</td>
<td>SYS005</td>
</tr>
<tr>
<td>SORT4</td>
<td>SYS005</td>
<td>SYS006</td>
</tr>
<tr>
<td>IDMSDBL4</td>
<td>SYS006</td>
<td></td>
</tr>
</tbody>
</table>
Examples

The following example directs the MAINTAIN INDEX utility to rebuild the SKILL-NAME-NDX in the EMPDEMO segment.

```
maintain index in segment empdemo
  using empss01
  rebuild "skill-name-ndx" from members;
```

The following command directs MAINTAIN INDEX to rebuild the SKILL-NAME-NDX, to run all steps as a sortexit, and to reuse workfiles:

```
maintain index in segment empdemo using empss01
  as sortexit reuse workfiles
  rebuild skill-name-ndx from members;
```

Sample Output

The following listing is generated after successful completion of the MAINTAIN INDEX utility statement in the previous example.

```
IDMSBCF mm/dd/yy PAGE 1
IDMS Batch Command Facility

MAINTAIN INDEX IN SEGMENT EMPDEMO USING EMPSS01
  REBUILD "SKILL-NAME-NDX" FROM MEMBERS ;
UT010002 BEGINNING PROCESSING FOR STEP IDMSTABX
UT012001 IDMSTABX RELEASE nn.n TAPE volser STARTED
UT012004 69 INTERMEDIATE RECORDS WERE WRITTEN TO SYS003
UT012006 SYS003 MAXIMUM RECORD SIZE IS 78
UT012007 IDMSTABX RELEASE nn.n PROCESSING COMPLETED
UT010003 STEP IDMSTABX HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP SORT3
```
MERGE ARCHIVE

The MERGE ARCHIVE utility is used to merge the archived journal files of data sharing group members that are sharing update access to data. It can also be used to merge archive and local mode journal files to simplify a subsequent recovery operation.

The output file created by the merge utility can be used as input to the ROLLFORWARD, ROLLBACK, EXTRACT JOURNAL, and MERGE ARCHIVE utility statements.

The merge utility also produces a report identifying global quiesce points within the set of merged journal images.

Under certain circumstances, this utility must be used if members of a data sharing group are sharing update access to data.

⚠️ **Note:** For more information about when the use of this utility is mandated, see Backup and Recovery in the *CA IDMS Database Administration Section.*

This article describes the following information:

- **Authorization** (see page 142)
- **Syntax** (see page 142)
- **Parameters** (see page 142)
- **Usage** (see page 143)
- **JCL Considerations** (see page 144)
Authorization

<table>
<thead>
<tr>
<th>To</th>
<th>You Need This Privilege</th>
<th>On</th>
</tr>
</thead>
<tbody>
<tr>
<td>merge archived journal files</td>
<td>USE</td>
<td>The DMCL</td>
</tr>
</tbody>
</table>

Syntax

```
MERGE ARCHIVE ──┬── COMPLETE ──┬── INCOMPLETE ──┘
              └─ REPORT ──┬── FULL ──┐
                └─ TERSE ──┐
                       └─ SUMMARY ──┘
```

Parameters

- **COMPLETE**
  Indicates that all journal files that contain images needed for recovery have been included as input. To complete transactions that are incomplete at the end of the process, additional checkpoint records will be written to the merged output file:

  - For incomplete distributed transactions whose state is InDoubt, additional distributed and local checkpoint records will be written if a matching section recovery control input file entry is provided.
  - For local (that is, non-distributed) transactions still active at the end of the process, an ABRT checkpoint record will be written.

  The output file may then be used as input to the ROLLBACK, ROLLFORWARD, or EXTRACT JOURNAL utility functions.

- **INCOMPLETE**
  Indicates that only a subset of the journal files have been included as input. The journal files are merged, but no additional checkpoints are generated. The output file can be used only as input to an EXTRACT JOURNAL or a subsequent MERGE ARCHIVE. You should not use the output file as input to ROLLBACK or ROLLFORWARD.

- **REPORT**
  Specifies the amount of detail that is to appear on the report produced by the merge utility.

- **FULL**
  Specifies that all details are to be reported. This includes for every transaction: checkpoints, database statistics, and area usage. All details of a distributed transaction record are reported.
This includes local transaction ids with program names, external transaction ids, and resource manager interests. Additionally, transactions active at the end of the process are listed and the time of the last global quiesce point is identified. FULL is the default if no REPORT option is specified.

- **TERSE**
  Indicates that only transaction checkpoints and summary information is produced. For distributed transaction records: external transaction ids and resource manager interests are not included in the report.

- **SUMMARY**
  Indicates that only final summary information is produced.

**Usage**

**Processing flow**

MERGE ARCHIVE uses two input files: SYS001 and JRNM01 and one output file: SYS002. It proceeds by sorting the contents of SYS001 in chronological sequence and then merging the results of the sort with the contents of JRNM01. The resulting merged journal images are then processed and written to SYS002. Control records are also written to SYS002 indicating the range of images for each member that are included on the merged file and an indication of whether or not the output file was created with the COMPLETE option.

**Input files**

SYS001 is used to supply input that is not in chronological sequence. Typically, archive files from one or more data sharing members are concatenated together as input to SYS001, although it is also possible to include one or more merged files as input. The order of concatenation is not relevant.

JRNM01 is used to supply a single merged archive file. Multiple files cannot be concatenated as input to JRNM01. If no merged archive file exists, then JRNM01 should be specified as dummy.

**Incremental merging**

To minimize recovery time, journal files can be merged periodically and the output from each merge operation used as input to a subsequent MERGE ARCHIVE.

When merging journal files incrementally, specify the INCOMPLETE option on every MERGE ARCHIVE execution except the final one. The final merge operation before executing a ROLLFORWARD or ROLLBACK utility statement must specify the COMPLETE option.

**Using disk files**

It is possible to use disk files for merged journal files. This can be beneficial for incremental merging, since all intermediate merged files can reside on disk. Only the final merged file (the one created with the COMPLETE option) may need to be written to tape so that ROLLFORWARD or ROLLBACK can process it.

**Incomplete transactions**
If a transaction is encountered whose initial checkpoint record (BGIN) is not contained on the input files being processed, a warning message is written that will result in a return code of 4. This is not necessarily an error, since missing journal images can be merged at a later time; however, the missing journal records might need to be provided before the merged file can be used for recovery purposes. For more information, see the ROLLFORWARD or ROLLBACK utility commands.

**MERGE ARCHIVE and Distributed Transactions**

MERGE ARCHIVE reports on distributed transactions and supports the use of input and output section recovery control files. If COMPLETE is specified, the input section recovery control file is used to complete InDoubt distributed transactions. If an output section recovery control file is included in the JCL, an entry will be written for each incomplete distributed transaction encountered. For more information, see JCL Considerations (see page 144) and the "Common Facilities for Distributed Transactions" section.

⚠️ **Note:** For considerations associated with distributed transactions during recovery operations, see the *CA IDMS Database Administration Section*.

### JCL Considerations

When submitting a MERGE JOURNAL statement through the batch command facility, in addition to the standard JCL required for the batch command facility, you must also include statements to define:

- SYS001 to point to the concatenated set of archived journal files or merged journal files
- JRNM01 to point to a single merged journal file. If no such file exists, JRNM01 must be specified as DUMMY
- SYS002 to point to the merged output file. The output file will have the block size that is specified for the archive journal file in the DMCL used for the merge
- Any sort work files needed by your local sort

To use a section recovery input control file, include a CTRLIN file definition or DD statement in the IDMSBCF execution JCL. To use a section recovery output control file, include a CTRLOUT file definition or DD statement in the IDMSBCF execution JCL. The format of these files is fixed block with a record length of 80.

⚠️ **Note:** For more information about the generic JCL used to execute the batch command facility, see the section for your operating system in this section.
Example

The following statement directs the MERGE ARCHIVE utility to create a merged output file of all archived journal records and to write ABRT checkpoint records for any transaction still active when all input has been processed.

```
merge archive complete;
```

Sample Output

The following is output generated after submitting a MERGE ARCHIVE statement to the batch command facility.

```
MERGE ARCHIVE COMPLETE REPORT TERSE ;
NODE SYSTEM72 RU_ID 46 PGM_ID DBCRPUD QUIESCE LEVELS 1 UPD 0 BGIN yyyy-mm-dd-hh.mm.ss.ffffff GLBQ
NODE SYSTEM74 RU_ID 42 PGM_ID DBCRPUD QUIESCE LEVELS 1 UPD 0 BGIN yyyy-mm-dd-hh.mm.ss.ffffff GLBQ
NODE SYSTEM74 RU_ID 43 PGM_ID DBCRPUD QUIESCE LEVELS 2 UPD 1 BGIN yyyy-mm-dd-hh.mm.ss.ffffff GLBQ
NODE SYSTEM74 RU_ID 45 PGM_ID DBCRPUD QUIESCE LEVELS 3 UPD 2 BGIN yyyy-mm-dd-hh.mm.ss.ffffff GLBQ
NODE SYSTEM72 RU_ID 46 PGM_ID DBCRPUD QUIESCE LEVELS 0 UPD 0 ENDJ yyyy-mm-dd-hh.mm.ss.ffffff GLBQ
NODE SYSTEM72 RU_ID 50 PGM_ID DBCRPUD QUIESCE LEVELS 1 UPD 0 BGIN yyyy-mm-dd-hh.mm.ss.ffffff GLBQ
NODE SYSTEM72 RU_ID 51 PGM_ID DBCRPUD QUIESCE LEVELS 2 UPD 1 BGIN yyyy-mm-dd-hh.mm.ss.ffffff GLBQ
NODE SYSTEM72 RU_ID 52 PGM_ID DBCRPUD QUIESCE LEVELS 3 UPD 2 BGIN yyyy-mm-dd-hh.mm.ss.ffffff GLBQ
NODE SYSTEM72 RU_ID 53 PGM_ID DBCRPUD QUIESCE LEVELS 4 UPD 3 BGIN yyyy-mm-dd-hh.mm.ss.ffffff GLBQ
NODE SYSTEM74 RU_ID 42 PGM_ID DBCRPUD QUIESCE LEVELS 2 UPD 2 ENDJ yyyy-mm-dd-hh.mm.ss.ffffff GLBQ
NODE SYSTEM74 RU_ID 49 PGM_ID DBCRPUD QUIESCE LEVELS 1 UPD 0 BGIN yyyy-mm-dd-hh.mm.ss.ffffff GLBQ
...

ACTIVE PROGRAMS AT STOP TIME WERE:
NONE

DATABASE IN QUIESCE AT END OF FILE
DATABASE IN UPDATE QUIESCE AT END OF FILE
DATA BASE MAY NOT NEED TO BE RECOVERED
SYS001 BLOCK COUNT 40 RECORD COUNT 6194
JRNM01 BLOCK COUNT 342 RECORD COUNT 52235
SYS002 BLOCK COUNT 382 RECORD COUNT 58429
```

More Information

- For more information about section recovery, see the CA IDMS Database Administration Section.
- For more information about data sharing, see the CA IDMS System Operations Section.
PRINT INDEX

The PRINT INDEX utility reports on the structure of system-owned indexes and indexed sets. Using the PRINT INDEX utility, you can review:

- The number of levels in an index
- The contents of the fixed and variable portions of one or more SR8 records in an index
- The amount of available space on the page containing each SR8 in an index

This article describes the following information:

- Authorization (see page 146)
- Syntax (see page 146)
- Parameters (see page 147)
- Usage (see page 150)
- JCL Considerations (see page 151)
- Examples (see page 151)
- Sample Output (see page 152)

Authorization

<table>
<thead>
<tr>
<th>To</th>
<th>You Need This Privilege</th>
<th>On</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report on indexes in a segment</td>
<td>DBAREAD</td>
<td>The area containing the index and the area(s) containing records referenced by the index</td>
</tr>
</tbody>
</table>

Syntax

```
PRINT INDEX set-name set-specifications 
SR8 occurrence-key-1 report-option 
ONLY TREE FULL LEG SUMMARY 
NEXT PRIOR LVL level-number 
DECIMAL HEX TERSE 
SEGMENT segment-name 
DBNAME db-name 
USING subschema-name 
OWNER occurrence-key-2 
```
Parameters

- **set-name**
  Specifies the name of the system-owned index or indexed set on which the PRINT INDEX statement is to report.
  When processing a system-owned index, processing begins at the first SR8 record in the SR7-SR8 set.

- **SEGMENT**
  Specifies the segment containing the index structures to be reported. When using the FULL option, or when specifying a starting member dbkey, the member area must also exist in this segment.

- **segment-name**
  Specifies the name of the segment.

- **DBNAME**
  Specifies the database containing the index structures to be reported.

- **db-name**
  Specifies the name of the database.

- **USING subschema-name**
  Specifies the name of the subschema in which the named indexed set is included.

- **TABLE schema.table-id**
  Specifies the name of a table.
- **REFERENCED ROWID**
  For the named table, directs the PRINT INDEX utility to report on the index occurrence whose
  owner is the referenced row identified by *occurrence-key*.

- **REFERENCING ROWID**
  For the named table, directs the PRINT INDEX utility to report on the index occurrence containing
  the row ID of the referencing row identified by *occurrence-key*.

- **'x'hex-database-key'**
  Specifies the hexadecimal database key of an owner or member record in the specified indexed
  set.

- **page-num**
  Specifies the page number of an owner or member record in the specified indexed set.

- **line-num**
  Specifies the line number of an owner or member record in the specified indexed set.

- **SR8**
  Identifies the index to be processed by specifying an index of an SR8 record in the index.

- **page-group**
  Identifies the page group of the SEGMENT where the index resides.

- **'x'database-key'**
  Specifies the hexadecimal database key of the SR8 record.

- **page-num**
  Specifies the page number of the SR8 record.

- **line-num**
  Specifies the line number of the SR8 record.

- **ONLY**
  Directs the PRINT INDEX utility to report only on the SR8 record used as the entry point into the
  index. ONLY is the default when you do not specify a portion of the index structure to report on.

- **TREE**
  Directs the PRINT INDEX utility to report on all the SR8 records in the index, starting with the top-
  level SR8. SR8s are processed by following the next pointers.

- **FULL**
  Directs the PRINT INDEX utility to report on:
  - All the SR8 records in the index, starting with the top-level SR8. SR8s are processed by
    following the next pointers.
  - The database key, index pointer value, and orphan condition of each member record in the
    index. Member records are processed by walking the bottom level of the index.
LEG
Directs the PRINT INDEX utility to report on the SR8 records connected by up pointers, starting with the SR8 used as the entry point into the index.
For an unsorted index or for an entry SR8 that is the top-level SR8 in a sorted index, specifying LEG has the same effect as specifying ONLY.

NEXT
Directs the PRINT INDEX utility to report on the SR8 records connected by next pointers in a single level of the index, starting with the SR8 used as an entry point into the index.

PRIOR
Directs the PRINT INDEX utility to report on the SR8 records connected by prior pointers in a single level of the index, starting with the SR8 used as an entry point into the index.

LVL
Directs the PRINT INDEX utility to report on all the SR8 records in a single level of the index.

level-number
Specifies the index level to report on; an integer in the range 0 through 255.
By default, if you do not specify an index level, the PRINT INDEX utility reports on the SR8s in the level of the SR8 record used as the entry point into the index.

SUMMARY
Requests a summary report for the target index. A summary report consists of three parts:

- Part 1 (header) provides general information on the index definition.
- Part 2 (main body) provides information on index owner occurrence(s). A system-owned index contains a single index owner; a user-owned index can contain more than one index owner.
- Part 3 (index overview) provides global statistical information for a user-owned index only.

A summary report on a system-owned index contains parts 1 and 2.
A summary report on a user-owned index always contains parts 1 and 3. Part 2 is included only in a detailed summary report.

- ONLY
Requests a summary report with parts 1 and 3 for the target user-owned index. This parameter is ignored for a system-owned index. ONLY is the default.

- DETAILED
Requests a summary report with parts 1, 2, and 3 for the target user-owned index. This parameter is ignored for a system-owned index.

DECIMAL
Directs the PRINT INDEX utility to print both the fixed and variable portions of each SR8 record in the report. Symbolic keys in the variable portion of each SR8 are printed in decimal (display) format.
DECIMAL is the default when you do not specify the way in which the contents of the SR8s in the index are to be printed.
**Usage**

**How to submit the PRINT INDEX statement**

You submit the PRINT INDEX statement by using the batch command facility or the online command facility.

**When to use PRINT INDEX**

The PRINT INDEX utility can help you determine whether an index needs to be rebuilt. For example, you should consider rebuilding an index when the PRINT INDEX utility report on the index indicates one of the following:

- The number of index levels is greater than anticipated for the original index structure.
- Twenty-five percent or more of the member records are orphans.

An index can be rebuilt using MAINTAIN INDEX or TUNE INDEX. For more information about index rebuilding and indexing in general, see the *CA IDMS Database Administration Section*.

**Note:** The output of PRINT INDEX without the SUMMARY parameter is proportional to the number of index members that are being reported. If PRINT INDEX is run online or in batch through CV, the output is buffered in scratch. If the scratch area cannot contain all the output, PRINT INDEX fails with a task abend.

**Hexadecimal display of symbolic keys**

The HEX parameter of the SET/SR8 statement is useful when the symbolic key for the index is a non-displayable data type, such as binary or packed.

**When to use DBNAME**

You can use DBNAME instead of SEGMENT at any time. You must use it when an index member resides in a different segment from the index structure, and the FULL option is used, or you specify a starting MEMBER dbkey.
JCL Considerations

When you submit a PRINT INDEX utility through the batch command facility, the JCL to execute the facility must include statements to define:

- The database files that contain the indexes and member records to be accessed

⚠️ **Note:** For more information about the generic JCL used to execute the batch command facility, see the section for your operating system in this section.

Examples

**Printing an entire index**

The following example directs the PRINT utility to report on the EMP-IDX-SET using the FULL option.

```plaintext
PRINT INDEX "EMP-IDX-SET" DBNAME VLDBDBN USING VLDBSUBC FULL;
```

**Printing the bottom level of an index**

The following example directs the PRINT utility to report on the COV-IDX-SET using the LEVEL and TERSE options.

```plaintext
PRINT INDEX "COV-IDX-SET" SEGMENT VLDBSPG1 USING VLDBSUBC MEMBER X'01390448' LVL 0 TERSE;
```

**Printing individual SR8 record**

The following example directs the PRINT utility to report on a specific SR8 record.

```plaintext
PRINT INDEX SR8 5:80130:03 NEXT 2 HEX;
```

**Printing an index from an SQL-defined database**

The following example directs the PRINT utility to report on the EMP-COVERAGE index that is part of the SQLSPG.EMPLOYEE table.

```plaintext
PRINT INDEX "EMP-COVERAGE" SEGMENT VLDBSPG1 TABLE SQLSPG.EMPLOYEE SUMMARY;
```

**Printing a summary report of an index**

The following example directs the PRINT utility to report on the DEPT_EMPL index using the SUMMARY option.

```plaintext
PRINT INDEX DEPT_EMPL SEGMENT USERDB TABLE DEMO.DEPT SUMMARY;
```

**Printing a REFERENCING ROWID summary report of an index**
The following example directs the PRINT utility to report on the index occurrence containing the row ID of the referencing row identified by X'01390201'.

```
PRINT INDEX "COV-IDX-SET" SEGMENT VLDBSPG1 TABLE SQLSPG.
COVERAGE REFERENCING ROWID X'01390201' SUMMARY;
```

**Sample Output**

**Printing an entire index**

The PRINT INDEX utility generates the following report after successful completion of the statement in the previous "Printing an Entire Index" example.

```
IDMSBCF 18.0 CA IDMS Batch Command Facility mm/dd

PRINT INDEX "EMP-IDX-SET" DBNAME VLDBDBN
 USING VLDBSUBC FULL;
SET=EMP-IDX-SET OWNER=SR7
PAGE GROUP=2 RECORDS PER PAGE=255
0DBK=01394301 SR8 01394303 SR8 01394306 ASC CUSH=12 SYM TKL=3 COMP
01394301 01394309 01394302 01394305 01394307 01394309 0139430A 0139430B 0139430C 0139430D 0139430E 0139430F 01394310 01394311 01394312 01394313 01394314 01394315 01394316 01394317 01394318 01394319 0139431A 0139431B 0139431C 0139431D 0139431E 0139431F 01394320 01394321 01394322 01394323 01394324 01394325 01394326 01394327 01394328 01394329 0139432A 0139432B 0139432C 0139432D 0139432E 0139432F 01394330 01394331 01394332 01394333 01394334 01394335 01394336 01394337 01394338 01394339 0139433A 0139433B 0139433C 0139433D 0139433E 0139433F 01394340 01394341 01394342 01394343 01394344 01394345 01394346 01394347 01394348 01394349 0139434A 0139434B 0139434C 0139434D 0139434E 0139434F 01394350 01394351 01394352 01394353 01394354 01394355 01394356 01394357 01394358 01394359 0139435A 0139435B 0139435C 0139435D 0139435E 0139435F 01394360 01394361 01394362 01394363 01394364 01394365 01394366 01394367 01394368 01394369 0139436A 0139436B 0139436C 0139436D 0139436E 0139436F 01394370 01394371 01394372 01394373 01394374 01394375 01394376 01394377 01394378 01394379 0139437A 0139437B 0139437C 0139437D 0139437E 0139437F 01394380 01394381 01394382 01394383 01394384 01394385 01394386 01394387 01394388 01394389 0139438A 0139438B 0139438C 0139438D 0139438E 0139438F 01394390 01394391 01394392 01394393 01394394 01394395 01394396 01394397 01394398 01394399 0139439A 0139439B 0139439C 0139439D 0139439E 0139439F 013943A0 013943A1 013943A2 013943A3 013943A4 013943A5 013943A6 013943A7 013943A8 013943A9 013943AA 013943AB 013943AC 013943AD 013943AE 013943AF 013943B0 013943B1 013943B2 013943B3 013943B4 013943B5 013943B6 013943B7 013943B8 013943B9 013943BA 013943BB 013943BC 013943BD 013943BE 013943BF 013943C0 013943C1 013943C2 013943C3 013943C4 013943C5 013943C6 013943C7 013943C8 013943C9 013943CA 013943CB 013943CC 013943CD 013943CE 013943CF 013943D0 013943D1 013943D2 013943D3 013943D4 013943D5 013943D6 013943D7 013943D8 013943D9 013943DA 013943DB 013943DC 013943DD 013943DE 013943DF 013943E0 013943E1 013943E2 013943E3 013943E4 013943E5 013943E6 013943E7 013943E8 013943E9 013943EA 013943EB 013943EC 013943ED 013943EE 013943EF 013943F0 013943F1 013943F2 013943F3 013943F4 013943F5 013943F6 013943F7 013943F8 013943F9 013943FA 013943FB 013943FC 013943FD 013943FE 013943FF
```

**Printing the bottom level of an index**

The next report illustrates the use of the LVL and TERSE options to request the printing of the bottom level of an index.
Printing individual SR8 records

The following report illustrates the use of the SR8 option to request the printing of a specific SR8 record.

Printing an index from an SQL-defined database

The following example provides a report, using the FULL option, on an SQL-defined index.
Printing a summary report of an index

The following report illustrates the use of the SUMMARY option to request the printing of a user-owned index.

```
PRINT INDEX 'EMP-COVERAGE' SEGMENT VLDSPG1
TABLE SQLSPG.EMPLOYEE SUMMARY;
SET Name: EMP-COVERAGE
IBC 70 Displacement 0
Sort option NOT SORTED Key length N/A
Duplicates FIRST Compression No
OWNER: EMPLOYEE
AREA VLDSPG1.EMPL-AREA Low page (SUB: 80056
Page size 4276 High page AREA) 80100
Page group 5 Records per page 255
MEMBER: COVERAGE Set membership Optional Automatic
Located VIA index Yes Displ’t 0
AREA VLDSPG1.COVE-AREA Low page (SUB- 80106
Page size 4276 High page AREA) 80150
Page group 5 Records per page 255

Index overview
Nr of owner occurrences 56
Nr of owner occurrences 56
Nr of empty owners 55 98.2%
Nr of displaced top level SR8s 0 0.0%
Nr of SR8s: Total 2
Average 0.0
Highest 2 Owner X'0138D404'
Min. nr of SR8s: Total 2
Average 0.0
Highest 2 Owner X'0138D404'
Nr of levels: Average 0.0
Highest 1 Owner X'0138D404'
Min. nr of levels: Average 0.0
Highest 1 Owner X'0138D404'
Nr of pages: Average 0.0
Highest 1 Owner X'0138D404'
Min. nr of pages: Average 0.0
Highest 1 Owner X'0138D404'
Nr of occurrences with orphans 0
Nr of Orphans: Total 0 0.0%
Highest 0 Owner *** N/A ***
Total size of all SR8s 368
Size of largest SR8 316

Distribution of Index Levels
....+....20...+....40...+....60...+....80...+....
  2+| 0 0.0%
  1 | 1 1.7%
  0 |*************************************************| 55 98.2%

Distribution of Minimum Index Levels
....+....20...+....40...+....60...+....80...+....
  2+| 0 0.0%
  1 | 1 1.7%
  0 |*************************************************| 55 98.2%

Distribution of Number of SR8s
....+....20...+....40...+....60...+....80...+....
  3+| 0 0.0%
  2 | 1 1.7%
  1+| 0 0.0%
```
Printing a REFERENCING ROWID summary report of an index

The following report illustrates the use of the REFERENCING ROWID option to request the printing of the index occurrence containing the row ID of the referencing row identified by X'01390201'.

```
PRINT INDEX "COV IDX-SET" SEGMENT VLDBSPG1
    TABLE SQLSPG.COVERAGE
    REFERENCING ROWID X'01390201' SUMMARY;
```

| Status = 0        | SQLSTATE = 00000 |

OWNER: SR7

AREA: VLDBSPG1.COVE-AREA

Page size 4276

Page group 5

Records per page 255

MEMBER: COVERAGE

Set membership Optional Automatic

Located VIA index No

Index is Linked

AREA: VLDBSPG1.COVE-AREA

Page size 4276

Page group 5

Records per page 255

OWNER X'01390201' on page 80130

Top level SR8 on page 80130

utilization 5.7%

Index occurrence totals

Nr of members 74
Report Output Description

- **Part 1 -- Header**
  The report header provides general information on the index definition, the index owner record or SQL table, and the index member record or SQL table.

- **Part 2 -- Details for each index occurrence**
  A detailed report on index run-time data per index occurrence is always output for a system-owned index. For a user-owned index, it is output only when explicitly requested using SUMMARY DETAILED. The report provides the following:
  - The DBKEY of the index owner record occurrence and its page number.
  - The page number of the first (top level) SR8. Ideally, the top level SR8 should reside on the same page as the index owner, except for an index with only one level and a non-zero index displacement.
  - The number of entries that are used in the top level SR8 is expressed as a percentage utilization of the maximum IBC count assigned to the index.
  - At the intermediate and bottom level (output only if the index occurrence has more than 1 level):
    - Number of SR8's and its computed minimum value
    - Number of pages with SR8's and its computed minimum value
    - Number of displaced SR8's and as a percentage of SR8's
    - Number of entries in use and as a percentage of available entries
    - Number of orphans and as a percentage of used entries
    - Total size of all SR8's
  - Index occurrence totals:
    - Number of levels in the index and its computed minimum value
    - Number of members in the index
    - Size of the largest SR8
- Number of SR8's and its computed minimum value
- Number of pages with SR8's and its computed minimum value
- Number of displaced SR8's and as a percentage of SR8's
- Number of entries in use and as a percentage of available entries
- Number of orphans and as a percentage of used entries
- Total size of all SR8's

- Estimated IO's versus number of database buffers for sequential bottom level access indicates the physical "sequentiality" of the index. Ideally, the number of I/O's should not vary with the number of buffers and should be equal to the number of pages with bottom level SR8's.

A displaced SR8 is a bottom level SR8 located within the index displacement or a non-bottom level SR8 located outside the index displacement. A computed minimum value is obtained by using the current number of entries in the index, filling SR8's to 100% using the current value of INDEX BLOCK CONTAINS for the index, and assuming that all space on a database page is available to hold the index owner and the associated SR8's.

**Part 3 -- Index overview and distribution diagrams for a user-owned index**

- Index overview
  An index overview provides the following information:
  - Number of owner occurrences
  - Number of empty owners and as a percentage of owner occurrences
  - Number of displaced (not on same page as owner) top level SR8's
  - Total, average, and highest value of the number of SR8's
  - Total, average, and highest value of the computed minimum number of SR8's
  - Average and highest value of the index level
  - Average and highest value of the computed minimum level
  - Average and highest values of the computed minimum number of pages
  - Number of index occurrences with orphans
  - Number of orphans: total and as a percentage of the number of entries and highest plus its owner DBKey
  - Total size of all SR8's
  - Size of largest SR8
Distribution diagrams
A distribution diagram provides the number and percentage of index occurrences for a certain property in both a numeric and a pseudo-graphical way. Properties for which a distribution diagram is output are:

- Index level
- Minimum index level
- Number of SR8's
- Number of members in the index occurrence
- Estimated IOs using 1 buffer for sequential bottom level access
- Number of pages with intermediate level SR8's
- Minimum number of pages with intermediate level SR8's
- Percentage displaced intermediate level SR8's
- Number of pages with bottom level SR8's
- Minimum number of pages with bottom level SR8's
- Percentage displaced bottom level SR8's

More Information

- For more information about designing indexes, see the *CA IDMS Database Design Section*.
- For more information about defining and maintaining indexes, see the *CA IDMS Database Administration Section*.

PRINT JOURNAL

The PRINT JOURNAL utility reports on transaction checkpoints in an archive journal file.

This article describes the following information:

- Authorization (see page 159)
- Syntax (see page 159)
- Parameters (see page 159)
- Usage (see page 161)
- JCL Considerations (see page 161)
- Example (see page 162)
- Sample Output (see page 162)
Authorization

<table>
<thead>
<tr>
<th>To</th>
<th>You need this privilege</th>
<th>On</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report on checkpoints for a journal file</td>
<td>USE</td>
<td>The DMCL associated with the journal file</td>
</tr>
</tbody>
</table>

Syntax

```
PRINT JOURNAL

UTC

START AT 'date date-time'

STOP AT 'date date-time'

REPORT FULL TERSE SUMMARY
```

Parameters

- **START AT**
  
  Directs the PRINT JOURNAL utility to report only on checkpoints written for transactions that started on or after the indicated date and time. By default, if you do not specify a start date or date and time, processing begins with the first checkpoint written in the archive journal file.

- **STOP AT**
  
  Directs the PRINT JOURNAL utility to report on checkpoints written only for transactions that started on or before the indicated date and time. By default, if you do not specify a stop date or date and time, processing ends with the last checkpoint written to the archive journal file.

- **date**
  
  Specifies the date, in one of the following formats:

  - `yyyymm-dd`
  - `mm/dd/yyyy`

  In these formats, the following rules apply:

  - `yyyy` specifies the year. `yyyy` must be an integer in the range 0001 through 9999. Leading zeros are optional.
- **mm** specifies the month within the year. *mm* must be an integer in the range 01 through 12. Leading zeros are optional.

- **dd** specifies the day within the month. *dd* must be an integer in the range 01 through 31. Leading zeros are optional.

The combined values of *yyyy*, *mm*, and *dd* must represent a valid date. For example, 1988-02-29 is a valid date. 1989-02-29 is not.

- **date-time**
  Specifies the date and time, where:

  - The format for specifying the date and time are:
    \[ yyyy-mm-dd-mm.mm.ssffffff \]

  - The rules for specifying the date component of DATE-TIME are the same as for the DATE option described previously. Rules for specifying the TIME component are:
    - **Hh** specifies the hour on a 24-hour clock. *hh* must be an integer in the range 00 through 23. Leading zeros are optional.
    - **mm** specifies the number of minutes past the hour. *mm* must be an integer in the range 00 through 59. Leading zeros are optional.
    - **ss** specifies the number of seconds past the minute. *ss* must be an integer in the range 00 through 59. Leading zeros are optional.
    - **ffffff** specifies the number of millionths of a second past the specified second. *ffffff* is optional; if you include it, it must be an integer in the range 000000 through 999999. The default value is 000000. Trailing zeros are optional.

- **time**
  Specifies the time, in the following format:

  - \[ hh:mm:ss \]

  The rules for specifying *time* are the same as those listed for DATE-TIME. When specifying *time*, the date defaults to the current date.

- **REPORT**
  Specifies the amount of detail that is to appear on the report.

- **FULL**
  Specifies that all details are to be reported. For every transaction, this includes checkpoints, database statistics, and area usage. All details of a distributed transaction record are reported. This includes local transaction ids with program names, external transaction ids, and resource manager interests. Additionally, transactions active at the end of the process are listed and the time of the last global quiesce point is identified. FULL is the default if no REPORT option is specified.
• **TERSE**
  Indicates that only transaction checkpoints and summary information is produced. For distributed transaction records: external transaction ids and resource manager interests are not included in the report.

• **SUMMARY**
  Indicates that only final summary information is produced.

• **UTC**
  Specifies that Start and Stop times are interpreted as UTC times instead of local times.

**Usage**

**How to submit the PRINT JOURNAL statement**

You submit the PRINT JOURNAL statement only through the batch command facility. When submitting PRINT JOURNAL statements, you must run the batch command facility in local mode.

**PRINT JOURNAL and Distributed Transactions**

PRINT JOURNAL reports on distributed transactions and supports the use of input and output section recovery control files. The input section recovery control file is used to complete InDoubt distributed transactions. If an output section recovery control file is included in the JCL, an entry will be written for each incomplete distributed transaction encountered. For more information, see JCL Considerations (see page 161) and the "Common Facilities for Distributed Transactions" section.

**Note:** For considerations associated with distributed transactions during recovery operations, see the *CA IDMS Database Administration Section*.

**JCL Considerations**

When you submit a PRINT JOURNAL statement to CA IDMS/DB through the batch command facility, the JCL to execute the facility must include statements to define the archive journal file.

To use a manual recovery input control file, include a CTRLIN file definition or DD statement in the IDMSBCF execution JCL. To use a manual recovery output control file, include a CTRLOUT file definition or DD statement in the IDMSBCF execution JCL. The format of both of these files is fixed block with a record length of 80.

**Note:** For more information about the generic JCL used to execute the batch command facility, see the section for your operating system in this section.
Example

The following example directs the PRINT JOURNAL utility to report on all transaction checkpoints in the archive journal file beginning with the first checkpoint.

```
print journal;
```

Sample Output

The PRINT JOURNAL utility produces the following report.

```
IDMSBCF         IDMS Batch Command Facility            mm/dd/yy PAGE 1
PRINT JOURNAL;
RU_ID 1 PGM_ID EMPLOAD QUIESECE LEVELS 1 UPD 0 BEGIN yyyy-mm-dd hh.mm.ssffffff
RU_ID 1 PGM_ID EMPLOAD QUIESECE LEVELS 0 UPD 0 ENDJ yyyy-mm-dd hh.mm.ssffffff
STATISTICS FOR EMPLOAD RU_ID 1
PAGES READ 969 PAGES WRITTEN 847 PAGES REQUESTED 2567 CALC TARGET 186
CALC OVERFLOW 0 VIA TARGET 439 VIA OVERFLOW 0 LINES REQUESTED 6042
RECS CURRENT 1307 CALLS TO IDMS 1461 FRAGMENTS STORED 0 RECS LOCATED 0
LOCKS REQUESTED 0 SELECT LOCKS 0 UPDATE LOCKS 0
START TIME: yyyy-mm-dd hh.mm.ssffffff TIME OF LAST COMMIT: NONE NUMBER OF COMMITS: 0
SINCE LAST COMMIT

TABLESPACES OPENED BEFORE AFTER BEFORE AFTER USAGE MODE
EMPDEMO.ORG-DEMO-REGION 825 825 825 825 SHARED UPDATE
EMPDEMO.INS-DEMO-REGION 115 115 115 115 SHARED UPDATE
EMPDEMO.EMP-DEMO-REGION 1314 1314 1314 1314 SHARED UPDATE

BLOCK COUNT 702 RECORD COUNT 5512
DATABASE IN QUIESCE AT END OF FILE
DATABASE IN UPDATE QUIESCE AT END OF FILE
ACTIVE PROGRAMS AT STOP TIME WERE:
NONE
DATA BASE MAY NOT NEED TO BE RECOVERED
Status = 0
AutoCommit will COMMIT transaction
Command Facility ended with no errors or warnings
```

Note: For more information about journaling and transaction checkpoints, see the CA IDMS Database Administration Section.

PRINT LOG

The PRINT LOG utility prints all or selected portions of the DC/UCF system log or an archive log file created by the ARCHIVE LOG statement.

You can select portions for printing based on:
This article describes the following information:

- Authorization (see page 163)
- Syntax (see page 163)
- Parameters (see page 164)
- Usage (see page 166)
- JCL Considerations (see page 166)
- Examples (see page 166)
- Sample Output (see page 167)

### Authorization

<table>
<thead>
<tr>
<th>To print a system or archive log</th>
<th>You Need This Privilege</th>
<th>On</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DBAREAD</td>
<td>The SYSTEM.DDLCLOG area of the database associated with the DC/UCF system whose log you want to print</td>
</tr>
</tbody>
</table>

### Syntax

```
PRINT LOG

FROM DATABASE ARCHIVE

DUMPS MESSAGES STATISTICS TRACES

UTC

FOR terminal-id

START AT 'date date-time time'

STOP AT 'date date-time time'
```
Parameters

- **FROM**
  Specifies whether print log information is to be printed from the SYSTEM.DDLDCLOG area or from an archive log file.

- **DATABASE**
  Prints log information from the SYSTEM.DDLDCLOG area.

- **ARCHIVE**
  Prints log information from an archive log file.

- **DUMPS**
  Includes snap dumps in the printed information.

- **MESSAGES**
  Includes DC/UCF system messages in the printed information.

- **STATISTICS**
  Includes DC/UCF system statistics in the printed information.

- **TRACES**
  Includes user traces in the printed information.

**Note:** If you specify one or more of DUMPS, MESSAGES, STATISTICS, and TRACES, only the specified type of information is printed. If you do not specify any of these options, all types of information are printed.

- **FOR**
  Prints only log information associated with one or more specified logical or physical terminals. By default, if you do not specify FOR, log information associated with all logical or physical terminals is printed.

- **terminal-id**
  - **logical** -- The identifier of a logical terminal defined to the DC/UCF system.
  - **physical** -- The identifier of a physical terminal defined to the DC/UCF system. This parameter will direct the system to select only SYSTEM STATISTICS PTERM records.

  You can specify up to 32 logical or physical terminals.

- **START AT**
  Prints only log information recorded at or after the specified time. By default, if you do not specify START AT, information from the beginning of the system log or archive log file is printed.
### STOP AT

Prints only log information recorded at or before the specified time.

By default, if you do not specify STOP AT, all information recorded in the system log or archive log file (starting at the time specified in the START parameter, if any) is printed.

### date

Specifies the date, in one of the following formats:

- `yyyy-mm-dd`
- `mm/dd/yyyy`

In these formats, the following rules apply:

- `yyyy` specifies the year. `yyyy` must be an integer in the range 0001 through 9999. Leading zeros are optional.
- `mm` specifies the month within the year. `mm` must be an integer in the range 01 through 12. Leading zeros are optional.
- `dd` specifies the day within the month. `dd` must be an integer in the range 01 through 31. Leading zeros are optional.

The combined values of `yyyy`, `mm`, and `dd` must represent a valid date. For example, 1988-02-29 is a valid date. 1989-02-29 is not.

### date-time

Specifies the date and time, where:

- The format for specifying the DATE-TIME is:
  
  `yyyy-mm-dd-hh.mm.ss.ffffff`

- The rules for specifying the DATE component of DATE-TIME are the same as for DATE described previously. The rules for specifying the TIME component of DATE-TIME are:

  - `hh`: specifies the hour on a 24-hour clock. `hh` must be an integer in the range 00 through 23. Leading zeros are optional.
  - `mm`: specifies the number of minutes past the hour. `mm` must be an integer in the range 00 through 59. Leading zeros are optional.
  - `ss`: specifies the number of seconds past the minute. `ss` must be an integer in the range 00 through 59. Leading zeros are optional.
  - `ffffff`: specifies the number of millionths of a second past the specified second. `ffffff` is optional; if you include it, it must be an integer in the range 000000 through 999999. The default value is 000000. Trailing zeros are optional.

### time

Specifies the time in the following format:

- `hh:mm:ss`
The rules for specifying TIME are the same as those listed for DATE-TIME shown previously. When specifying only the TIME option, the date defaults to the current date.

- **UTC**
  Specifies that Start and Stop times are interpreted as UTC times instead of local times. If coded, all times the utility writes are also displayed as UTC times instead of local times.

### Usage

#### How to submit the PRINT LOG statement

You submit a PRINT LOG statement only through the batch command facility. You can run the batch command facility only in local mode.

#### When to use PRINT LOG

Use the PRINT LOG utility only when the system log is being written to the DDLDCLOG area of the data dictionary.

#### When not to use PRINT LOG

If the system log is assigned to one or more sequential files, you should use the appropriate operating system utility (for example, IEBGENER for z/OS systems or DITTO for z/VSE systems) to print the contents of the log file.

### JCL Considerations

When you submit a PRINT LOG statement through the batch command facility, the JCL to execute the facility must include statements to define:

- The system log area (DDLDCLOG) if you specify FROM DATABASE
- The archive log file whose contents you want to print if you specify FROM ARCHIVE
- The system message area (DDLDCMSG)
- The dummied journal file

⚠️ **Note:** For more information about the generic JCL used to execute the batch command facility, see the section for your operating system.

### Examples

*Printing everything except dumps*
The following PRINT LOG statement requests printing of all messages, statistics, and user traces currently in the data dictionary log area.

```c
print log from database
  messages
  statistics
  traces;
```

### Printing all information for a specified time period

The following PRINT LOG statement requests printing of all information recorded in the system log from January 18, 1999, at 8:00 p.m. until just before 3:00 a.m. on January 19, 1999. The PRINT LOG utility will retrieve this information from the archive log file.

```c
print log from archive
  start at '1999-1-18-20.00.00'
  stop at '1999-1-19-02.59.59.999999';
```

### Printing user traces for a logical terminal

The following PRINT LOG statement requests printing of all user traces for logical terminal LTVT0M5 beginning at 2:30 p.m. on the current day. The PRINT LOG utility will retrieve this information from the data dictionary log area.

```c
print log from database
  traces
    for ltvtm05
  start at '14.30.00';
```

### Sample Output

The following PRINT LOG report requests the printing of all information recorded in the system log from September 19, 1999, at 8:10 a.m. until just after 2:20 p.m. on September 19, 1999. The PRINT LOG utility will retrieve this information from the archive log file.

```
<table>
<thead>
<tr>
<th>IDMSBCF</th>
<th>IDMS Batch Command Facility</th>
<th>09/19/99</th>
<th>PAGE 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRINT LOG FROM DATABASE START AT '1999-09-19-08.10.00.00000'</td>
<td>STOP AT '1999-09-19-14.20.10.00000';</td>
<td></td>
<td></td>
</tr>
<tr>
<td>*** PAGE 000830002 STATUS 0000 091699 19.14.59 6 S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*** PAGE 000831500 STATUS 0000 091699 10.19.58 6 S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*** PAGE 000831750 STATUS 0000 091699 14.30.01 6 S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*** PAGE 000831875 STATUS 0000 091699 16.46.34 6 S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*** PAGE 000831937 STATUS 0000 091699 17.54.59 6 S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*** PAGE 000831968 STATUS 0000 091699 18.34.58 6 S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*** PAGE 000831984 STATUS 0000 091699 18.54.59 6 S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*** PAGE 000831992 STATUS 0000 091699 19.84.58 6 S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*** PAGE 000831996 STATUS 0000 091699 19.10.00 6 S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*** PAGE 000831998 STATUS 0000 091699 19.14.58 6 S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*** PAGE 000831999 STATUS 0000 091699 19.14.59 6 S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*** PAGE 000832000 STATUS 0000 091699 19.14.59 6 S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*** PAGE 000830002 STATUS 0000 091699 19.14.59 6 S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*** PAGE 000832000 STATUS 0000 091699 19.14.59 6 S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*** PAGE 000831001 STATUS 0000 091799 14.20.00 6 S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*** PAGE 000831500 STATUS 0000 091699 10.19.58 6 S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*** PAGE 000831250 STATUS 0000 091799 16.42.43 6 S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### More Information

- For more information about viewing the contents of the DC/UCF system log online, see [CA IDMS System Tasks and Operator Commands](https://docops.ca.com/display/IDMSCU19/CA+IDMS+System+Tasks+and+Operator+Commands).

- For more information about defining the DC/UCF system log, see [CA IDMS System Generation](https://docops.ca.com/display/IDMS/CA+IDMS+System+Generation).

- For more information about maintaining the DC/UCF system log, see [Administrating CA IDMS System Operations](https://docops.ca.com/display/IDMS/CA+IDMS+System+Operations+V19+Next).

- For more information about statistics written to the DC/UCF system log, see [DC/UCF System Reports](https://docops.ca.com/pages/viewpage.action?pageId=309110511).

### PRINT PAGE

The PRINT PAGE utility prints the contents of one or more database pages in display (decimal) and/or hexadecimal format. You can request printing of:

<table>
<thead>
<tr>
<th>volser</th>
<th>Release nn.n</th>
<th>Licensed from CA</th>
<th>DATE</th>
<th>TIME</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>091999 09.23.57</td>
<td>IDMS DC013002 V72 T8 ATTACHING DATABASE RESOURCE CONTROLLER</td>
<td></td>
<td>09/19/99</td>
<td>09:26:36</td>
<td>2</td>
</tr>
<tr>
<td>091999 09.23.57</td>
<td>IDMS DC020013 V72 T1 Lock Manager Initialization Complete</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>091999 09.23.57</td>
<td>IDMS DC200007 V72 T2 3005 Error Accessing Area - Not Available. SYSDEF.DDLDCLSCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>091999 09.23.57</td>
<td>IDMS DC201801 V72 T1 CA IDMS/DB: 72 Started</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>091999 09.23.57</td>
<td>IDMS DC00009 V72 T1 CA IDMS/DB Active 09:23:41 91.262</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>091999 09.23.57</td>
<td>IDMS DC013003 V72 T0 OPENING SYSTEM RUN UNITS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>091999 09.23.57</td>
<td>IDMS DC013014 V72 T0 ATTACHING TASK FOR SERVICE DRIVER RHDCRUSD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>091999 09.23.57</td>
<td>IDMS DC013014 V72 T0 ATTACHING TASK FOR SERVICE DRIVER RHDCRUSD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>091999 09.23.57</td>
<td>IDMS DC013014 V72 T0 ATTACHING TASK FOR SERVICE DRIVER RHDCRUSD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>091999 09.23.57</td>
<td>IDMS DC013014 V72 T0 ATTACHING TASK FOR SERVICE DRIVER RHDCRUSD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>091999 09.23.57</td>
<td>IDMS DC013014 V72 T0 ATTACHING TASK FOR SERVICE DRIVER RHDCRUSD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>091999 09.23.57</td>
<td>IDMS DC050001 V72 T0 DCLOG IS 00% FULL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>091999 09.24.03</td>
<td>IDMS DC021010 V72 LOAD OF RESIDENT MODULE RHDCUX28 FAILED -- CODE 20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>091999 09.24.03</td>
<td>IDMS DC021010 V72 LOAD OF RESIDENT MODULE F100D11S FAILED -- CODE 20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>091999 09.24.04</td>
<td>IDMS DC013016 V72 T0 SECURITY SYSTEM INITIALIZATION BEGINNING</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>091999 09.24.04</td>
<td>SYSTEM INITIALIZED ON 91262 AT 09:24:04.35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>091999 09.24.04</td>
<td>MAP OF REGION</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>091999 09.24.04</td>
<td>RHDCO590 0006E68 IDMSDB10 0002F030 OPT 00045870</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>091999 09.24.04</td>
<td>CSA 00045D70 CCE 0004E560 SCAREA 00050280 CSAAREA 00050700</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>091999 09.24.04</td>
<td>DDT 00053340 LTT 00053380 PTT 00053380</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>091999 09.24.04</td>
<td>TDT 0005B428 PTT 0005B560 TCEBUFS 0005B60 TCA 0005B70</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>091999 09.24.04</td>
<td>DCSAREA 00055280 TCSAREA 0005E580 MMPDTBL 0013DC00 ECLLIST 0013D300</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>091999 09.24.04</td>
<td>RCA 0013E640 RLEAREA 0013E6A0 RCEAREA 0014D3E8 OPERAREA 001679C8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>091999 09.24.04</td>
<td>RCEAREA 0016CDE8 SCF 0016D060 STGPOOL 00173000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>091999 09.24.04</td>
<td>RENTPOOL 00180000 RHDCO64W 001D200 RHDCO5S 001D300 RHDCO1B 001D300</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>091999 09.24.04</td>
<td>RHDCO6E 001D600 RHDCO2U 001E400 RHDCO1V 001E600 STGPOOL 0045E00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>091999 09.24.04</td>
<td>XALODBUF 0065E00 ABENDSTG 00665000 HIADSTG 00666320 HLT 00960A8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

16-Jan-2018 168/493
The target page for a specified CALC key

A specified range of pages

All or some of the pages in an area or subarea

The following topics are discussed on this page:

- **Authorization (see page 169)**
- **Syntax (see page 169)**
- **Parameters (see page 170)**
- **Usage (see page 172)**
- **JCL Considerations (see page 172)**
- **Example (see page 172)**
- **Sample Output (see page 173)**

### Authorization

<table>
<thead>
<tr>
<th>To</th>
<th>You Need This Privilege</th>
<th>On</th>
</tr>
</thead>
<tbody>
<tr>
<td>Print one or more pages in an area</td>
<td>DBAREAD</td>
<td>The area</td>
</tr>
</tbody>
</table>

### Syntax

```
PRINT PAGE

PAGE-GROUP: start-page TO end-page FOR page-count pages
IN AREA segment.area
SUBAREA subarea FOR page-count pages
FOR CALC key calc-options

DISPLAY
HEX
BOTH

Expansion of calc-options

'character-key-value'

X'hex-key-value'

IN AREA segment-name.area-name

SUBAREA subarea-name
CALC start-page TO end-page

FOR page-count pages
OVERFLOW
```
Parameters

- **page-group:**
  Specifies the page group from which one or more pages is to be printed. Page group is concatenated with the *start-page* in the form `page-group:start-page`.
  By default, if you do not specify a page group, page group zero is used.

- **start-page**
  The number of the page to print or the first in a range of pages to print.

- **TO**
  Specifies a range of pages to be printed.

- **end-page**
  The last page of a range of pages to be printed. *End-page* must be greater than or equal to *start-page*.

- **FOR page-count pages**
  Specifies that the indicated number of database pages is to be printed.
  *Page-count* must be an integer in the range 1 through 32,768.
  If you specify a number higher than the remaining number of pages in the area, printing continues with the first page of the area. Processing stops when the specified number of pages are printed or when all the pages in the area are printed, whichever comes first. No pages are printed more than once.
  By default, if you do not specify TO or FOR, one page is printed.

- **IN AREA**
  Specifies pages from a specified area to be printed.
  By default, if you specify IN AREA, but do not specify SUBAREA, all the pages in the specified area are printed.

- **segment**
  Specifies the name of the segment associated with the area whose pages are to be printed.

- **area**
  Specifies the name of the area whose pages are to be printed.

- **SUBAREA**
  Specifies that only the pages in a subarea are to be printed.

- **subarea**
  Specifies the name of the subarea whose pages are to be printed.

- **FOR page-count pages**
  Specifies the indicated number of database pages, starting with the first page of the (sub)area, to be printed.
  *Page-count* must be an integer in the range 1 through 32,768.
  If you specify a number higher than the number of pages in the (sub)area, processing will stop when all the pages in the (sub)area are printed.
  By default, if you do not specify the FOR clause, all pages in the area or subarea are printed.
FOR CALC key
Specifies that one or more pages based on CALC keys will be printed.
Note: You cannot use the FOR CALC option for locating the target page of a large key value, because you can only specify key values that are less than or equal to 256 bytes.

calc-options
Specifies the CALC keys on which to base the selection of pages.

DISPLAY
Directs the PRINT PAGE utility to print the contents of the requested database pages in display format only.
By default, if you do not specify DISPLAY or HEX, the contents of the specified pages will be printed in both display and hexadecimal format.

HEX
Directs the PRINT PAGE utility to print the contents of the requested database pages in hexadecimal format only.

BOTH
Directs the PRINT PAGE utility to print the contents of the requested database pages in both display and hexadecimal format.
BOTH is the default.

'character-key-value'
Specifies a CALC key with a character string literal. The target page in the specified area is printed.

X'hex-key-value'
Specifies a CALC key with a hexadecimal literal. The target page in the specified area is printed.

IN AREA
Identifies the area from which the target page for the specified CALC key is to be printed.
If you specify neither SUBAREA nor CALC in calc-options, the page range of the area specified by IN AREA is used to determine the target page for the specified CALC key.

segment-name
Specifies the name of the segment associated with the area containing the target page to be printed.

area-name
Specifies the name of the area that contains the target page to be printed.

SUBAREA
Identifies the subarea of the area to be used in determining the target page for the specified CALC key.

subarea-name
Specifies the name of the subarea.

CALC
Identifies a page range of the area to be used in determining the target page for the specified CALC key.
**start-page**
Specifies the number of the first page in the page range.

**TO**
Identifies the end of the page range.

**end-page**
Specifies the number of the last page in the page range.

**FOR page-count pages**
Specifies the indicated number of database pages to be printed, starting with the target page for the specified CALC key.

Page-count must be an integer in the range 1 through 32,768.
If you specify a number higher than the remaining number of pages in the (sub)area, printing will continue with the first page of the (sub)area. Processing will stop when the specified number of pages is printed or when all the pages in the (sub)area are printed, whichever comes first. No pages are printed more than once.
By default, if you do not specify FOR or OVERFLOW, one page is printed.

**OVERFLOW**
Directs the PRINT PAGE utility to print (in addition to the target page) all pages in the specified (sub)area that contain records in the CALC chain of the target page for the specified CALC key due to overflow situations or that contain duplicates of the CALC key.

**Usage**

**How to submit the PRINT PAGE statement**
You submit the PRINT PAGE statement by using either the batch command facility or the online command facility. When submitting the PRINT PAGE utility through the batch command facility, you must run the batch command facility in local mode.

**JCL Considerations**
When you submit a PRINT PAGE statement through the batch command facility in local mode, the JCL to execute the facility must include statements to define the files containing the pages to be processed. To run under central version include a SYSCTL statement.

⚠️ **Note:** For more information about the generic JCL used to execute the batch command facility, see the section for your operating system in this section.

**Example**

Print a specific page in an area
The following example directs PRINT PAGE to print page 75020.

```
print page 75020;
```

Sample Output

The following report lists the contents of page 75020 in response to the PRINT PAGE statement in the previous example.

```
000000 001250C0 01250C01 01250C01 0F700000 ...........................
000010 01250C00 01250C00 01250B01 01250B01 ...........................
000020 01250B01 0124F002 0125A003 0125A001 ...........................
000030 01256702 01256701 0125C002 0125C002 ...........................
000040 0125C003 0125C003 0125C001 0125C001 ...........................
000050 01250317 01250317 0F04F057 0F04F057 ...........................
000060 E8040404 040040C9 040040C9 040040C9 ...........................
000070 40040404 407F574F E240405E 05E340E2 ...........................
000080 E39C5D7 04004040 40D5C1E3 09C35D24 ...........................
000090 40040404 40040404 40040404 40040404 ...........................
0000A0 F004F000 0F04F000 0F04F000 0F04F000 ...........................
0000B0 01250C01 01250C01 0125130F 0125BD05 ...........................
0000C0 0125BD05 F7F7F1F2 F0F6F0F1 00000000 ...........................
0000D0 F0F0F0F0 F0F0F0F0 F0F0F0F0 F0F0F0F0 ...........................
```

Status = 0
```
SET BATCH WIDTH PAGE 80;
PRINT PAGE 75020;
```

More Information

- For more information about defining CALC keys, see CA IDMS Database Administration Section.
For more information about database pages, see *CA IDMS Database Administration Section*.

**PRINT SPACE**

The PRINT SPACE utility reports on space utilization in one or more areas, subareas, or segments.

You can specify whether the report is to be based on information in the space management pages (SMPs) or on information in the database page headers.

This article describes the following information:

- Authorization (see page 174)
- Syntax (see page 174)
- Parameters (see page 174)
- Usage (see page 176)
- JCL Considerations (see page 176)
- Examples (see page 176)
- Sample Output (see page 177)

**Authorization**

You need DBAREAD privilege to access the following report functions:

- Report on space utilization in an area
- Report on space utilization in a segment
- Report on space utilization in a file

**Syntax**

```
PRINT SPACE

FOR
 AREA segment-name.area-name SUBAREA subarea-name
 SEGMENT segment-name
 FILE segment-name.file-name

FULL
```

**Parameters**

- **FOR**
  Identifies the areas on which the PRINT SPACE utility is to report.
• **AREA**
  Directs the PRINT SPACE utility to report on space utilization in one or more areas or subareas. If no SUBAREA clause is specified, this option produces a report for the entire area plus a report for each file in the area. If a SUBAREA clause is specified, reporting is restricted to the specified subarea.

  **Note:** Native VSAM files are ignored if used in this utility.

  • **segment-name**
    Specifies the name of the segment associated with the area.

  • **area-name**
    Specifies the name of the area.

  • **subarea-name**
    Specifies the name of the subarea associated with the area.

• **SEGMENT**
  Directs the PRINT SPACE utility to report on space utilization in all areas of one or more segments.

  **Note:** Native VSAM files are ignored if used in this utility.

  • **segment-name**
    Specifies the name of the segment.

• **FILE**
  Directs the PRINT SPACE utility to report on space utilization for each area or portion of an area contained in the file. This option always produces a full report, whether you specify the FULL parameter.

  • **segment-name**
    Specifies the name of the segment associated with the file.

  • **file-name**
    Specifies the name of the file.

• **FULL**
  Directs the PRINT SPACE utility to base the space utilization report on information in the header of each page.
  By default, if you do not specify FULL, the space utilization report is based on information in the SMPs of the specified areas.

  **Note:** If the FULL option is not selected, the space utilization report is based solely on information on the space management pages (SMPs). The space utilization reported may vary widely from the actual space utilization as SMP statistics are not altered until the page referenced is at least 70% full.
Usage

Not using the FULL option

If the FULL option is not specified, the space utilization report is based on information on the Space Management Pages. The information reported may vary widely from the actual space utilization. For example, if all pages in a particular segment were 50% full, the SMP pages will indicate that each page is 100% available. When using the default on the PRINT SPACE utility, the report will indicate that the database is being used at a 0% level. When running the PRINT SPACE utility with the FULL option, the report will indicate that your database is 50% utilized.

Running Under Central Version

Only PRINT SPACE FOR AREA and PRINT SPACE FOR SEGMENT are supported under central version.

Logically-deleted records and reports

PRINT SPACE BY FILE sequentially reads the files, letting you include only the files in the JCL stream that you want to process.

When you use this option, PRINT SPACE will not report relocated logically deleted records as logically deleted. These records will be reported as normal records. Therefore, record space utilization reports for an area can produce different results when compared to the file report for the same page range.

JCL Considerations

When you submit a PRINT SPACE statement through the batch command facility in local mode, the JCL to execute the facility must include statements to define the file(s) containing the area(s) to be processed. When running under central version a SYSCTL statement is needed.

⚠️ Note: For more information about the generic JCL used to execute the batch command facility, see JCL and Commands for z/OS and z/VSE (https://docops.ca.com/pages/viewpage.action?pageId=357362200).

Examples

PRINT SPACE by area with FULL option

print space for area empdemo.emp-demo-area full;

PRINT SPACE by file

print space for file empdemo.empdemo;
Sample Output

The following report is produced by the PRINT SPACE utility after processing the PRINT SPACE statement by area with the FULL option.

IDMSBCF mm/dd/yy PAGE 1
IDMS Batch Command Facility

SET BATCH WIDTH PAGE 80;
Status = 0
PRINT SPACE FOR AREA EMPDEMO.EMP-DEMO-AREA FULL;

AVAILABLE Space Distribution Report

<table>
<thead>
<tr>
<th>AREA</th>
<th>EMPDEMO.EMP-DEMO-AREA</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGE SIZE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>512</td>
</tr>
<tr>
<td>PAGES</td>
<td>1,310,001 THRU 1,310,010</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Space</th>
<th>NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVAILABLE</td>
<td></td>
</tr>
<tr>
<td>91-100%</td>
<td>7</td>
</tr>
<tr>
<td>81-90 %</td>
<td>2</td>
</tr>
<tr>
<td>71-80 %</td>
<td>0</td>
</tr>
<tr>
<td>61-70 %</td>
<td>0</td>
</tr>
<tr>
<td>51-60 %</td>
<td>0</td>
</tr>
<tr>
<td>41-50 %</td>
<td>0</td>
</tr>
<tr>
<td>31-40 %</td>
<td>0</td>
</tr>
<tr>
<td>21-30 %</td>
<td>0</td>
</tr>
<tr>
<td>11-20 %</td>
<td>0</td>
</tr>
<tr>
<td>00-10 %</td>
<td>0</td>
</tr>
<tr>
<td>SMPS</td>
<td>1</td>
</tr>
<tr>
<td>TOTAL</td>
<td>10</td>
</tr>
</tbody>
</table>

FILE EMPDEMO.EMPDEMO

PAGES 1,310,001 THRU 1,310,010
BLOCKS 1 THRU 10

Total Space Allocated 5,120
Total Space Available (Percent) 4,128 (80%)
Total Space Used 992

AREA EMPDEMO.EMP-DEMO-REGION Distribution of USED Space Report

<table>
<thead>
<tr>
<th>Record Type</th>
<th>Length</th>
<th>Occurrences</th>
<th>Total Space Used</th>
<th>Percent of Total Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>SR1002</td>
<td>40</td>
<td>3</td>
<td>120</td>
<td>12.09</td>
</tr>
<tr>
<td>**LD1002</td>
<td>24</td>
<td>3</td>
<td>72</td>
<td>7.25</td>
</tr>
<tr>
<td>Space Inv.</td>
<td>480</td>
<td>1</td>
<td>480</td>
<td>48.38</td>
</tr>
<tr>
<td>Overhead</td>
<td>32</td>
<td>10</td>
<td>320</td>
<td>32.25</td>
</tr>
</tbody>
</table>

*** logically deleted records FOUND ***

Status = 0

The following report is produced by the PRINT SPACE utility after processing the PRINT SPACE statement by file.

IDMSBCF nn.n mm/dd/yy PAGE 1
CA IDMS Batch Command Facility
PRINT SPACE FOR FILE EMPDEMO.EMPDEMO;

AVAILABLE Space Distribution Report

<table>
<thead>
<tr>
<th>AREA</th>
<th>EMPDEMO.EMP-DEMO-REGION</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGE SIZE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4,276</td>
</tr>
<tr>
<td>PAGES</td>
<td>75,001 THRU 75,100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Space</th>
<th>NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVAILABLE</td>
<td></td>
</tr>
<tr>
<td>91-100%</td>
<td>78</td>
</tr>
<tr>
<td>81-90 %</td>
<td>13</td>
</tr>
<tr>
<td>71-80 %</td>
<td>7</td>
</tr>
<tr>
<td>61-70 %</td>
<td>0</td>
</tr>
<tr>
<td>51-60 %</td>
<td>0</td>
</tr>
<tr>
<td>41-50 %</td>
<td>0</td>
</tr>
<tr>
<td>31-40 %</td>
<td>1</td>
</tr>
</tbody>
</table>
Total Space Allocated 427,600
Total Space Available (Percent) 394,120 (92.17%)
Total Space Used 33,480
Logically Full Pages 0
Total Space Unusable (Percent) 0 (0.00%)

AREA EMPDEMO.EMP-DEMO-REGION Distribution of USED Space Report

<table>
<thead>
<tr>
<th>Record Type</th>
<th>Maximum</th>
<th>Length</th>
<th>Occurrences</th>
<th>Total Space Used</th>
<th>Percent of Total Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>SR7</td>
<td>40</td>
<td>40</td>
<td>1</td>
<td>40</td>
<td>0.11</td>
</tr>
<tr>
<td>SR8</td>
<td>1,396</td>
<td>2,820</td>
<td>3</td>
<td>2,820</td>
<td>8.42</td>
</tr>
<tr>
<td>SR415</td>
<td>196</td>
<td>10,976</td>
<td>56</td>
<td>10,976</td>
<td>32.78</td>
</tr>
<tr>
<td>SR420</td>
<td>60</td>
<td>4,080</td>
<td>68</td>
<td>4,080</td>
<td>12.18</td>
</tr>
<tr>
<td>SR425</td>
<td>36</td>
<td>5,400</td>
<td>150</td>
<td>5,400</td>
<td>16.12</td>
</tr>
<tr>
<td>SR460</td>
<td>40</td>
<td>2,720</td>
<td>68</td>
<td>2,720</td>
<td>8.12</td>
</tr>
<tr>
<td>Space Inv.</td>
<td>4,244</td>
<td>4,244</td>
<td>1</td>
<td>4,244</td>
<td>12.67</td>
</tr>
<tr>
<td>Overhead</td>
<td>32</td>
<td>3,200</td>
<td>100</td>
<td>3,200</td>
<td>9.55</td>
</tr>
</tbody>
</table>

*** NO logically deleted records found ***

Status = 0 SQLSTATE = 00000

Note: For more information about space utilization and database pages, see the Database Administration Section.
Total Unreserved Space Available (Percent) 2,729,240 (21.28%)
Total Space Used 9,587,632
Logically Full Pages 0
Total Space Unusable (Percent) 0 (0.00%)

AREA

Distribution ofUSED Space Report

Maximum

Record Type Length Occurrences Total Space Used Total Used
Null Line 8 54 432 0.00
SR5 48 1 48 0.00
SR7 40 12 480 0.00
SR8 1,404 28,567 9,482,184 99.93
Space Inv. 4,244 2 8,488 0.08
Overhead 32 3,000 96,000 0.98

*** NO logically deleted records found ***

PRINT TRACE

The PRINT TRACE utility prints all or selected trace entries that reside in a DDLDCTRC area, a DDLDCLOG area, or one or more archive files created by the ARCHIVE TRACE or ARCHIVE LOG utilities.

You can select entries for printing based on the type of information, date and time, and task ID.

This article describes the following information:

- Authorization (see page 179)
- Syntax (see page 179)
- Parameters (see page 180)
- Usage (see page 182)
- JCL Considerations (see page 182)
- Examples (see page 183)

Authorization

You need DBAREAD privilege to print trace information on the following system areas:

- SYSTEM.DDLDCTRC
- SYSTEM.DDLDCLOG

Syntax

The following diagram shows the syntax for the PRINT TRACE utility statement:

```
PRINT TRACE
FROM DATABASE ARCHIVE DDNAME SYS001 ddname
```
Parameters

This section describes the PRINT TRACE command parameters:

- **FROM**
  Indicates the location from which to print trace information: a database area or an archive file.

- **DATABASE**
  Prints trace information from either the DDLDCTR or the DDLDCLOG area. If the DMCL contains a DDLDCTR area, only trace information in that area is printed; otherwise, only trace information in the DDLDCLOG area is printed.

- **ARCHIVE**
  Prints trace information from one or more archive files.

- **DDNAME**
  Identifies the ddname of one or more archive files whose contents are to be printed. If more than one ddname is specified, trace information is merged and displayed in chronological sequence.

- **ddname**
  Specifies the ddname of an archive file.
  **Default:** SYS001
  **Limit:** 32 ddnames

  **Note:** Printing trace information from multiple archive files is only available to z/OS users. z/VSE users need to consolidate multiple archive files into a single file (in the order in which they were created) and use this file with the PRINT TRACE utility.

- **SYSTRACE**
  Includes basic system trace information in the output.

- **EXTENDED**
  Includes extended trace information in the output.
Note: If you specify SYSTRACE or EXTENDED, only the specified type of information is printed. If you do not specify either option, both types of information are printed.

- **TASKID**
  Prints trace information for selected tasks.
  
  - **taskid**
    Specifies the taskid of a task whose trace information is to be printed.
    Default: Trace information associated with all tasks is printed.
    Limit: 32 task identifiers.

- **START AT date-time**
  Prints only trace information recorded at or after the specified time.
  Default: Prints information from the beginning of the database area or archive file.

- **STOP AT date-time**
  Prints only trace information recorded at or before the specified date and time.
  Default: Prints information recorded in the database area or archive file (starting at the time specified in the START AT parameter, if any).

- **Expansion of date-time**
  
  - **date**
    Specifies a date, in one of the following formats:
    
    - yyyy-mm-dd
    - mm/dd/yyyy
    
    The date components are:
    
    - yyyy specified the year.
      Limit: 0001-9999 (leading zeros are optional)
    - mm specifies the month within the year.
      Limit: 01-12 (leading zeros are optional)
    - dd specifies the day within the month.
      Limit: 01-31 (leading zeros are optional)
    
    The combined values of yyyy, mm, and dd must represent a valid date. For example, 1988-02-29 is a valid date but 1989-02-29 is not.

  - **date-time**
    Specifies a date and time in one of the following formats:
    
    - yyyy-mm-dd-hh.mm.ss.ffffff
    - yyyy-mm-dd-hh.mm.ss
The date components are the same as those that can be specified for date.
The time components are:

- **hh** specifies the hour on a 24-hour clock.
  *Limit:* 00-23 (leading zeros are optional)

- **mm** specifies the number of minutes past the hour.
  *Limit:* 00-59 (leading zeros are optional)

- **ss** specifies the number of seconds past the minute.
  *Limit:* 00-59 (leading zeros are optional)

- **ffffff** specifies the number of millionths of a second past the specified second.
  *Limit:* 000000-999999 (trailing zeros are optional)

- **time**
  Specifies a time in the following format:

  - **hh:mm:ss**
    The time components are the same as those that can be specified for date-time.

- **UTC**
  Specifies that Start and Stop times are interpreted as UTC times instead of local times.

### Usage

**How to submit the PRINT TRACE statement**

You submit a PRINT TRACE statement only through the batch command facility. You can run the batch command facility only in local mode.

**When to use PRINT TRACE**

Use the PRINT TRACE utility when the system trace is being written to the DDLDCTRC area or DDLDCLOG area, if no DDLDCTRC area is defined in the dictionary.

### JCL Considerations

When you submit a PRINT TRACE statement through the batch command facility, the JCL to execute the facility must include statements to define the following:

- The system trace area (DDLDCTRC) or system log area (DDLDCLOG) if you specify FROM DATABASE

- The archive trace file whose contents you want to print if you specify FROM ARCHIVE

- The system message area (DDLDCMSG)

- The dummied journal file
Examples

Printing All Trace Entries

The following PRINT TRACE statement requests printing of all SYSTRACE and EXTENDED trace entries from the trace area.

PRINT TRACE FROM DATABASE;

Printing All Trace Information for a Specified Time Period

The following PRINT TRACE statement requests printing of all SYSTRACE and EXTENDED trace entries from the archive file, starting from March 02, 2011 at 4:12 p.m. until just before 4:30 p.m. on March 02, 2011 using the archive file with DDNAME SYS001.

PRINT TRACE FROM ARCHIVE
   DDNAME SYS001
   START AT '2016-03-02-21.12.00'
   STOP AT '2016-03-02-21.30.00';

Printing SYSTRACE Entries from an Archive File

The following PRINT TRACE statement requests printing of all SYSTRACE trace entries from the archive file with DDNAME SYS001.

PRINT TRACE FROM ARCHIVE
   DDNAME SYS001
   SYSTRACE;

Printing SYSTRACE and EXTENDED TRACE Entries from an Archive File

The following PRINT TRACE statement requests printing of all trace entries from the ARCHIVE file with DDNAME LOGARC.

PRINT TRACE FROM ARCHIVE
   DDNAME LOGARC;

PRINT TRACE Output for System Trace Entries

The following items are included in the PRINT TRACE output for system trace entries.

- **TOD TIME/DATE**
  The UTC time and date the trace entry was generated.

- **SCA**
  The relative SCA number of the subtask that generated the trace entry.

- **TASKID**
  Taskid of the task that generated the trace entry.

- **A(LTE)**
  The address of the generating task’s LTE.
### Output for Extended Entries

The following PRINT TRACE statement requests printing of all SYSTRACE and EXTENDED trace entries from the archive file starting from March 02, 2016 at 4:12 p.m. until just before 4:30 p.m. on March 02, 2016 using the archive file with DDNAME SYS001.

```
PRINT TRACE FROM ARCHIVE
DDNAME SYS001
START AT '2016-03-02-21.12.00'
STOP AT '2016-03-02-21.30.00';
```

### Sample Output

The following example shows the output from the Print Trace Utility:

```
PRINT TRACE;
CA IDMS DB/DC Print Trace Utility CA IDMS DB/DC is a Proprietary Software Product
CAGJ10 Release 1800 Licensed from CA, Inc.

*** DLDCTRC AREA FROM PAGES 000060001 TO 000062000
*** FIRST AND LAST PAGES SELECTED ARE 000060019 AND 000060041
*** Trace entries generated on: yyyy-mm-dd

TRACE ENTRIES, ORDERED OLDEST TO NEWEST.

TASKID/ TOD TIME/DATE SCA A(LTE) MOD MAC CALL R12/R3 R13/R4 R14/R5 R15/R6 R0/R7 R1/R8 TOD CLOCK
19.31.376391 00 122 292 22 13 80108200 38825560 36EC82AC 88BD5500 382B4CB8 00027350 0021820C C5699CD3C0807706
SYSTEM73 36EC8278 MTSY TSKCEP1 36EC9088 00020000 0021820C 36EC820C 3701CFBC 39C07000 00126E42
19.31.376391 00 122 5 240 1 80108200 382B4CB8 36EC8278 00020000 0021820C 36EC82D8 3701CFBC 39C07000 00126E42
SYSTEM73 36EC8278 MTSY TSKCEP1 36EC9088 00020000 0021820C 36EC820C 3701CFBC 39C07000 00126E42
19.31.376393 00 122 5 9 5 36EC8204 382B4CB8 36EC8278 00020000 0021820C 36EC82D8 3701CFBC 39C07000 00126E42
SYSTEM73 36EC8278 MTSY TSKCEP1 36EC9088 00020000 0021820C 36EC82D8 3701CFBC 39C07000 00126E42
19.31.376393 00 122 14 240 1 0000003C 382B4CB8 36EC8278 00020000 0021820C 36EC82D8 3701CFBC 39C07000 00126E42
```

---

**MOD**

The internal module number and four-character identifier of the module generating the trace entry.

**MAC**

The issuing macro number or entry point.

**CALL**

The relative macro expansion within the issuing program.

**R11/R2-R1/R8**

The contents of registers 11 through 1 and 2 through 8 at the time the entry was generated.

**TOD CLOCK**

The contents of the TOD clock at the time the entry was generated.

**<SYSTEM73>**

Node or data sharing member name of issuing system.
**PUNCH**

The PUNCH utility retrieves the DMCL or database name table load module from the dictionary and writes it, in object module form, into the SYSPCH file.

This article describes the following information:

- **Authorization** (see page 185)
- **Syntax** (see page 185)
- **Parameters** (see page 186)
- **Usage** (see page 186)
- **JCL Considerations** (see page 186)
- **Example** (see page 186)
- **Output** (see page 187)

### Authorization

<table>
<thead>
<tr>
<th>To punch</th>
<th>You need this privilege</th>
<th>For</th>
</tr>
</thead>
<tbody>
<tr>
<td>A DMCL</td>
<td>USE</td>
<td>The DMCL</td>
</tr>
<tr>
<td>A DBTABLE</td>
<td>USE</td>
<td>The DBTABLE</td>
</tr>
</tbody>
</table>

### Syntax

```
PUNCH

DMCL LOAD MODULE dmcl-name

DBTABLE LOAD MODULE dbtable-name
```
Parameters

- **DMCL LOAD MODULE**
  Directs the PUNCH utility to punch a DMCL load module from the dictionary.

- **DMCL-NAME**
  Specifies the load module name of the DMCL to be punched.

- **DBTABLE LOAD MODULE**
  Directs the PUNCH utility to punch a database name table load module from the dictionary.

- **DBTABLE-NAME**
  Specifies the load module name of the DBTABLE to be punched.

Usage

**How to submit the PUNCH statement**

You submit the PUNCH statement only through the batch command facility.

JCL Considerations

When you submit a PUNCH statement through the batch command facility, the JCL to execute the facility must include statements to define:

- The dictionary from which the load module is to be punched (local mode only)
- The journal file(s) associated with the DMCL (local mode only) (these can be dummied out)
- SYSPCH file.

**Note:** For more information about the generic JCL used to execute the batch command facility, see the section for your operating system in this section.

Example

The following example directs the PUNCH utility to retrieve the IDMSDMCL DMCL definition from the dictionary and write it, in object module form, to a SYSPCH file.

```
punch dmcl load module idmsdmcl;
```
Output

The CA IDMS Batch Command Facility returns the following listing after successful completion of the PUNCH utility.

PUNCH DMCL LOAD MODULE IDMSDMCL;
Status = 0

More Information

- For more information about defining DMCL and DBTABLE modules, see the CA IDMS Database Administration Section.
- For more information about deleting DMCL and DBTABLE load modules, see the CA IDMS Database Administration Section.

RELOAD

The RELOAD utility reloads a database using input created by the UNLOAD utility statement.

All areas in an SQL segment that are being used by a RELOAD must be either OFFLINE to any CV or in RETRIEVAL mode if the areas are not being UPDATED by the RELOAD.

⚠️ Note: You cannot use RELOAD to reload an SQL database that contains keys longer than 256 bytes. Instead, use the REORG utility to reorganize a large key database.

This article describes the following information:

- Authorization (see page 187)
- Syntax (see page 188)
- Parameter (see page 188)
- Usage (see page 189)
- JCL Considerations (see page 197)
- Example (see page 198)
- Sample Output (see page 199)

Authorization

To reload data into a segment, you need the privilege DBAWRITE on all areas associated with the segment.
Syntax

RELOAD

├─ STEP  ─┬─ step-name ┘
└─ FROM ┘

NOTIFY notify-record-count

as SORTEXIT

REUSE workfiles

Parameter

- **STEP**
  Specifies that only one step of the reload operation should be executed.
  If you do not specify STEP, all steps in the reload process are performed as a single operation.

- **step-name**
  Specifies the name of the step to execute.
  The name must be one of the following:
  - SORT1
  - IDMSDBL2
  - SORT2
  - IDMSDBLX
  - SORT3
  - IDMSDBL3
  - SORT4
  - IDMSDBL4

- **FROM**
  Specifies that RELOAD processing should begin at a specified step and complete all remaining steps.
  If you specify FROM, all remaining steps are treated together as a single operation.

- **step-name**
  Specifies the name of the step to execute.
  The name must be one of the following:
  - SORT1
  - IDMSDBL2
### Note
For more information and an explanation about when you can restart steps IDMSDBL2, IDMSDBL3, or IDMSDBL4, see Usage (see page 189).

- **NOTIFY**
  Indicates a message is sent to the system console after a specified number of records have been processed in the current step.

- **notify-record-count**
  Specifies the number of records to process before sending a message. By default or if 0 is specified, no message is sent to the system console except the standard message sent at the end of each step indicating the number of records processed during the step. Notify messages are only displayed by steps IDMSDBL2, IDMSDBL3, and IDMSDBL4 when a NOTIFY-RECORD-COUNT is specified.

- **as SORTEXIT**
  Causes each DBLx step in the utility to return its input data directly from the preceding sort instead of having the sort write the data to a workfile. This option eliminates one workfile for each sort and saves the I/O it takes to write, then read, the workfile.

- **REUSE workfiles**
  Causes each step in the utility to reuse an existing workfile, if possible, when writing its output data, instead of writing to a new one for each step. This reduces the number of workfiles that need to be allocated.

### Usage

**How to submit the RELOAD utility**

You submit the RELOAD utility only through the batch command facility. You must run the batch command facility in local mode.

**When to use RELOAD**

Use the RELOAD utility to reload a database that has been unloaded by UNLOAD.

When reloading an SQL-defined database, the database files must be formatted using the FILE option of the FORMAT utility statement so that AREA and TABLE stamps are properly reloaded.
When not to use RELOAD

Do not use the RELOAD utility to load a new non-SQL-defined database; use the FASTLOAD utility.

Do not use the RELOAD utility to load a new SQL-defined database; use the LOAD utility.

Run RELOAD all at once or in steps

The reloading process has eight steps, which you can run one at a time or all together. Each step generates output for use by the next step. Four of the steps are sorts to prepare data for use by the following step.

You can run each step separately, in which case you can use your own sorting program. Alternatively, you can direct the RELOAD utility to do the sorting for you. If you run the steps as a single process, the RELOAD utility will do the sorting for you automatically.

When to run RELOAD in steps

The most common reason to run the RELOAD utility in steps is to cut the work into pieces, each piece requiring less time to run than the whole process.

You could also decide to run the RELOAD utility in steps in order to use your own sorting programs between the steps, or you can run the sort steps on a different machine than the one holding the database.

Mixed Page Groups

RELOAD cannot process mixed page groups and will issue an error message if mixed page groups are encountered. You must use multiple invocations of the utility to process different page groups.

RELOAD and ASF databases

If the RELOAD utility is to be run against an ASF data or definition area, see Using ASF (https://docops.ca.com/display/IDMS19/Using+ASF).

Restarting IDMSDBL2, IDMSDBL3, or IDMSDBL4

If a problem arises while running IDMSDBL2 or IDMSDBL3, do not simply fix the cause of the problem and rerun the step. In addition to fixing the cause of the problem, you must do one of the following:

- Reinitialize the database and begin again at the IDMSDBL2 step
- If you backed up the database before running the step, you can run the step again, using the backup.

These steps change the database, and if a problem arises, you need to undo the changes before running a step over again.

If a non-data related problem occurs while running IDMSDBL4, you can restart the job without restoring. If the input file (SYS011) still exists, unlock the area and run the RELOAD utility from IDMSDBL4. Any updates previously performed by the run of IDMSDBL4 that abended will be overlayed by the updates in the restarted IDMSDBL4.
Data from UNLOAD is in SYS001, SYS002, SYS003, and RELDCTL

The information that the RELOAD utility needs is in the SYS001, SYS002, SYS003, and RELDCTL files. UNLOAD recognized these files as SYSPCH, SYS002, SYS003, and RELDCTL, respectively.

- The SYS001 file contains sort parameters.

  **Note:** If the RELOAD utility is started at a SORT step, in any mode, SYS001 should point to the sort parameters generated in a previous step.

- The SYS002 file contains the following:
  - Unloaded data from UNLOAD processing
  - Data to be loaded
  - Set membership information for each record

- The SYS003 file contains both system and user-owned index data from UNLOAD processing. If running in STEP mode, index data is not input until the SORT2 step.

- The RELDCTL file contains:
  - Set descriptor information
  - A control record containing subschema, segment, and DMCL information
  - Control information created during UNLOAD processing and used by RELOAD processing. If running in STEP mode, this file must be input to every step.

**SORTEXIT and FROM/STEP**

When using the FROM and STEP options with the SORTEXIT option, each pair of SORTn and DBLx steps are considered to be one step. If either half of the SORTn/DBLx is specified on a FROM or STEP option, processing will start with the SORTn step and the DBLx step will also be executed. For example:

- FROM IDMSDBL3 will start with step SORT3 and will continue to the end.
- STEP SORT3 will run steps SORT3 and IDMSDBL3.

**SORTEXIT/REUSE WORKFILE restart considerations**
Since SORTEXIT combines each SORTn step with the DBLx step that follows it, if a failure occurs in the DBLx step, a restart (if a restart is possible) must begin with the sort step and the input to the step will be resorted. Non-SORTEXIT mode will take longer to run but can be restarted after the sort in this case. Therefore, if restart time is more critical than normal runtime do not run the utility as a sortexit.

If the REUSE WORKFILE option is used with SORTEXIT, some input workfiles will be used as output files in the same step. Therefore, if these two options are used together and a failure occurs, the utility must be restarted from the beginning.

**Workfile considerations for restarting a failed RELOAD**

If the RELOAD command fails, depending on the reason for failure, restart the command at the failing step using the "FROM step-name" syntax. You can restart a step only if the input files to that step are intact and valid.

To prepare for a possible restart when running a one-step RELOAD, the intermediate work files should have a disposition that preserves the data set in the event of an abend, for example, "DISP=(NEW,CATLG,CATLG)."

To restart RELOAD at a particular step, the input files to that step must have a disposition to specify that the files already exist, for example, "DISP=OLD."

To determine which files were input to a given step see the "Intermediate Work File" tables under "JCL Considerations." Partially created output files should be deleted before you restart the job, and the original disposition should be used in the restart job, for example, "DISP=(NEW,CATLG,CATLG)."

The SYSPCH file contains sort parameter information for sort steps. It is an output file to IDMSDBLx steps but is not read unless restarting or running in step mode. So during a normal run the SYSPCH file should be treated as a normal output file, for example, "DISP=(NEW,CATLG,CATLG).” However, restarting is not as straightforward. If the previous job failed in an IDMSDBLx step, the SYSPCH file was an output file and should be deleted before restarting. But if the failure occurred in a SORTx step, the contents of the SYSPCH file should contain the same values that were input to the SORTx step. In this case the SYSPCH file should be preserved and defined as a SYS001 input file to the restart step.

When the SORTEXIT option is used, the SORTx and IDMSDBLx steps are combined. If a failure occurs in this mode, the SYSPCH file should normally be preserved and used as a SYS001 input file to the restart. However, there is a small window at the end of a IDMSDBLx step where the SYSPCH file is opened for output and new SORT parameters are written. If the job fails at this point, the entire SORTx/IDMSDBLx step must be restarted, but the SYSPCH file will not be valid as a SYS001 input file. In this case, the sort parameters must be recreated by hand or the job must be restarted at an earlier IDMSDBLx step if possible. One way to avoid this situation is to run in step mode when running SORTEXIT mode.

The RELDCTL data set is always an input file to the first step of a RELOAD whether being restarted or not.

**REUSE WORKFILE considerations**
Some tape volume management systems consider the reuse of a tape volume for second time output processing an error even in the same job and will not allow you to make this mistake. It results in rerunning the job over again without the REUSE option. You can sometimes avoid this by specifying a zero retention period for the tape output volume.

**Intermediate work files**

The following tables indicate which work files are created and read by the different utility steps depending on the use of the SORTEXIT and REUSE WORKFILE options.

<table>
<thead>
<tr>
<th>Step</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RELOAD: NOT sortexit mode and NOT reusing workfile</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SORT1</td>
<td>SYS002</td>
<td>SYS004</td>
</tr>
<tr>
<td>IDMSDBL2</td>
<td>SYS004</td>
<td>SYS005 SYS006</td>
</tr>
<tr>
<td>SORT2</td>
<td>SYS003 SYS006</td>
<td>SYS007</td>
</tr>
<tr>
<td>IDMSDBLX</td>
<td>SYS007</td>
<td>SYS008</td>
</tr>
<tr>
<td>SORT3</td>
<td>SYS005 SYS008</td>
<td>SYS009</td>
</tr>
<tr>
<td>IDMSDBL3</td>
<td>SYS009</td>
<td>SYS010</td>
</tr>
<tr>
<td>SORT4</td>
<td>SYS010</td>
<td>SYS011</td>
</tr>
<tr>
<td>IDMSDBL4</td>
<td>SYS011</td>
<td></td>
</tr>
<tr>
<td><strong>RELOAD: NOT sortexit mode and REUSING workfiles.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SORT1</td>
<td>SYS002</td>
<td>SYS004</td>
</tr>
<tr>
<td>IDMSDBL2</td>
<td>SYS004</td>
<td>SYS005 SYS006</td>
</tr>
<tr>
<td>SORT2</td>
<td>SYS003 SYS006</td>
<td>SYS004</td>
</tr>
<tr>
<td>IDMSDBLX</td>
<td>SYS004</td>
<td>SYS006</td>
</tr>
<tr>
<td>SORT3</td>
<td>SYS005 SYS006</td>
<td>SYS004</td>
</tr>
<tr>
<td>IDMSDBL3</td>
<td>SYS004</td>
<td>SYS005</td>
</tr>
<tr>
<td>SORT4</td>
<td>SYS005</td>
<td>SYS004</td>
</tr>
<tr>
<td>IDMSDBL4</td>
<td>SYS004</td>
<td></td>
</tr>
<tr>
<td><strong>RELOAD: SORTEXIT mode and NOT reusing workfiles.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SORT1/SORT2</td>
<td>SYS002</td>
<td>SYS005 SYS006</td>
</tr>
<tr>
<td>IDMSDBL2/SORT3</td>
<td>SYS003 SYS006</td>
<td>SYS008</td>
</tr>
<tr>
<td>SORT3/SORT4/IDMSDBL4</td>
<td>SYS005 SYS008</td>
<td>SYS010</td>
</tr>
<tr>
<td><strong>RELOAD: SORTEXIT mode and REUSING workfiles.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SORT1/SORT2/IDMSDBL2</td>
<td>SYS002</td>
<td>SYS005 SYS006</td>
</tr>
<tr>
<td>SORT2/IDMSDBLX</td>
<td>SYS003 SYS006</td>
<td>SYS006</td>
</tr>
<tr>
<td>SORT3/IDMSDBL3</td>
<td>SYS005 SYS006</td>
<td>SYS005</td>
</tr>
<tr>
<td>SORT4/IDMSDBL4</td>
<td>SYS005</td>
<td></td>
</tr>
</tbody>
</table>
How RELOAD works

The RELOAD utility consists of the following steps, which you can run separately or as a single operation. Each step uses the output from an earlier step and generates output for use by the next step.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Input</th>
<th>Output</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOR</td>
<td>Sorts the contents of SYS02.</td>
<td></td>
<td>SYS004 contains the sorted record information</td>
<td>SYS004 record length = SYS002 record length from UNLOAD</td>
</tr>
<tr>
<td>SOR T1</td>
<td></td>
<td>SYS001 contains sort parameters from UNLOAD SYSPCH file</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYS002 contains the record information from UNLOAD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IDM</td>
<td>Populates the database by means of an area sweepbut does not connect any sets.</td>
<td></td>
<td>SYS005 contains member descriptors for chained sets</td>
<td>SYS005 record length = 32 bytes</td>
</tr>
<tr>
<td>L2</td>
<td>Prints statistics on the records written to the database.</td>
<td></td>
<td>SYS006 contains member descriptors for user owned index sets</td>
<td>SYS006 record length = 32 bytes</td>
</tr>
<tr>
<td></td>
<td>Note: Overflow statistics may be fewer than expected. To improve performance during a reload, IDMSDBL2 uses the dbkey of the previously stored record as a 'direct' dbkey if the next record to be stored has the same new target page. This reduces the number of overflow conditions.</td>
<td></td>
<td>SYS007 contains statistic report</td>
<td></td>
</tr>
<tr>
<td>SOR</td>
<td>Sorts the contents of SYS003 and SYS006.</td>
<td></td>
<td>SYS001 contains sort</td>
<td></td>
</tr>
<tr>
<td>SOR T2</td>
<td></td>
<td></td>
<td>SYS007 record length = larger of RELOAD's</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>record length = SYS007 record length</td>
<td></td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Step Description</th>
<th>Input</th>
<th>Output</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IDM</strong> Establishes pointers for each index in which each record participates but does not build the indexes in the database.</td>
<td>- parameters from IDMSDBL2 SYSPCH*&lt;br&gt;- SYS003 contains index information from UNLOAD&lt;br&gt;- SYS006 contains member descriptors for user owned index sets from IDMSDBL2</td>
<td>- SYS007 contains the sorted contents of SYS003 and SYS006</td>
<td>SYS006 or UNLOAD's SYS003</td>
</tr>
<tr>
<td><strong>SOR</strong> Sorts the contents of SYS005 and SYS008. T3</td>
<td>- SYS001 contains sort parameters from IDMSDBL2 SYSPCH*&lt;br&gt;- SYS005 from IDMSDBL2&lt;br&gt;- SYS008 from IDMSDBLX</td>
<td>- SYS009 contains sorted index set descriptors</td>
<td>SYS009 record length = larger of RELOAD's SYS008 or IDMSDBL2's SYS005</td>
</tr>
<tr>
<td><strong>IDM</strong> Establishes pointers for each chained set in which each record participates (builds the record prefix) but does not write the prefixes to the database.</td>
<td>- SYS003 contains index information from UNLOAD&lt;br&gt;- SYS006 contains member descriptors for user owned index sets from IDMSDBL2</td>
<td>- SYS007 contains the sorted contents of SYS003 and SYS006</td>
<td>SYS006 or UNLOAD's SYS003</td>
</tr>
<tr>
<td><strong>IDM</strong> Establishes pointers for each chained set in which each record participates (builds the record prefix) but does not write the prefixes to the database.</td>
<td>- SYS003 contains index information from UNLOAD&lt;br&gt;- SYS006 contains member descriptors for user owned index sets from IDMSDBL2</td>
<td>- SYS007 contains the sorted contents of SYS003 and SYS006</td>
<td>SYS006 or UNLOAD's SYS003</td>
</tr>
<tr>
<td><strong>IDM</strong> Establishes pointers for each chained set in which each record participates (builds the record prefix) but does not write the prefixes to the database.</td>
<td>- SYS003 contains index information from UNLOAD&lt;br&gt;- SYS006 contains member descriptors for user owned index sets from IDMSDBL2</td>
<td>- SYS007 contains the sorted contents of SYS003 and SYS006</td>
<td>SYS006 or UNLOAD's SYS003</td>
</tr>
<tr>
<td><strong>IDM</strong> Establishes pointers for each chained set in which each record participates (builds the record prefix) but does not write the prefixes to the database.</td>
<td>- SYS003 contains index information from UNLOAD&lt;br&gt;- SYS006 contains member descriptors for user owned index sets from IDMSDBL2</td>
<td>- SYS007 contains the sorted contents of SYS003 and SYS006</td>
<td>SYS006 or UNLOAD's SYS003</td>
</tr>
<tr>
<td><strong>IDM</strong> Establishes pointers for each chained set in which each record participates (builds the record prefix) but does not write the prefixes to the database.</td>
<td>- SYS003 contains index information from UNLOAD&lt;br&gt;- SYS006 contains member descriptors for user owned index sets from IDMSDBL2</td>
<td>- SYS007 contains the sorted contents of SYS003 and SYS006</td>
<td>SYS006 or UNLOAD's SYS003</td>
</tr>
<tr>
<td><strong>IDM</strong> Establishes pointers for each chained set in which each record participates (builds the record prefix) but does not write the prefixes to the database.</td>
<td>- SYS003 contains index information from UNLOAD&lt;br&gt;- SYS006 contains member descriptors for user owned index sets from IDMSDBL2</td>
<td>- SYS007 contains the sorted contents of SYS003 and SYS006</td>
<td>SYS006 or UNLOAD's SYS003</td>
</tr>
<tr>
<td><strong>IDM</strong> Establishes pointers for each chained set in which each record participates (builds the record prefix) but does not write the prefixes to the database.</td>
<td>- SYS003 contains index information from UNLOAD&lt;br&gt;- SYS006 contains member descriptors for user owned index sets from IDMSDBL2</td>
<td>- SYS007 contains the sorted contents ofSYS003 and SYS006</td>
<td>SYS006 or UNLOAD's SYS003</td>
</tr>
<tr>
<td><strong>IDM</strong> Establishes pointers for each chained set in which each record participates (builds the record prefix) but does not write the prefixes to the database.</td>
<td>- SYS003 contains index information from UNLOAD&lt;br&gt;- SYS006 contains member descriptors for user owned index sets from IDMSDBL2</td>
<td>- SYS007 contains the sorted contents of SYS003 and SYS006</td>
<td>SYS006 or UNLOAD's SYS003</td>
</tr>
<tr>
<td><strong>IDM</strong> Establishes pointers for each chained set in which each record participates (builds the record prefix) but does not write the prefixes to the database.</td>
<td>- SYS003 contains index information from UNLOAD&lt;br&gt;- SYS006 contains member descriptors for user owned index sets from IDMSDBL2</td>
<td>- SYS007 contains the sorted contents of SYS003 and SYS006</td>
<td>SYS006 or UNLOAD's SYS003</td>
</tr>
<tr>
<td><strong>IDM</strong> Establishes pointers for each chained set in which each record participates (builds the record prefix) but does not write the prefixes to the database.</td>
<td>- SYS003 contains index information from UNLOAD&lt;br&gt;- SYS006 contains member descriptors for user owned index sets from IDMSDBL2</td>
<td>- SYS007 contains the sorted contents of SYS003 and SYS006</td>
<td>SYS006 or UNLOAD's SYS003</td>
</tr>
<tr>
<td><strong>IDM</strong> Establishes pointers for each chained set in which each record participates (builds the record prefix) but does not write the prefixes to the database.</td>
<td>- SYS003 contains index information from UNLOAD&lt;br&gt;- SYS006 contains member descriptors for user owned index sets from IDMSDBL2</td>
<td>- SYS007 contains the sorted contents of SYS003 and SYS006</td>
<td>SYS006 or UNLOAD's SYS003</td>
</tr>
</tbody>
</table>
### Step Description

<table>
<thead>
<tr>
<th>Step Description</th>
<th>Input</th>
<th>Output</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SOR</strong> Sorts the contents of SYS010.</td>
<td>SYS001 contains sort parameters from IDMSDBL3 SYSPCH*</td>
<td>SYS011 contains sorted prefix pointer information</td>
<td>SYS011 record length = same as SYS010 from prior step</td>
</tr>
<tr>
<td><strong>IDM</strong> Inserts the record prefixes by performing a serial SDB sweep of the database.</td>
<td>SYS011 from SORT4 RELDCTL file from UNLOAD utility</td>
<td>SYSLST contains messages on the results of the reload operation</td>
<td>-</td>
</tr>
</tbody>
</table>

* The sort parameters noted in the previous table are read from SYS001, only when you run the sort steps individually. If you run all steps together or all steps from an intermediate restart (using FROM), the sort parameters are passed in-storage. If you restart from an abnormal termination, point the SYS001 file to the SYSPCH file of the last completed step.

**Sort output after each step**

If you run the RELOAD utility a step at a time, you must use the sort parameters in the SYSPCH file to sort the contents of the intermediate work files.

Sort the intermediate work files as follows:

<table>
<thead>
<tr>
<th>Sort name</th>
<th>File to sort</th>
<th>Sort order</th>
<th>Sort on</th>
<th>Begins at</th>
</tr>
</thead>
<tbody>
<tr>
<td>SORT1</td>
<td>SYS002</td>
<td>Descending</td>
<td>20 bytes</td>
<td>Byte 5</td>
</tr>
</tbody>
</table>

---
CA IDMS Reference - 19.0

Sort name | File to sort          | Sort order | Sort on | Begins at |
----------|------------------------|------------|---------|-----------|
SORT2     | SYS003 and SYS006      | Ascending  | 24 bytes| Byte 5    |
SORT3     | SYS005 and SYS008      | Ascending  | 24 bytes| Byte 5    |
SORT4     | SYS010                 | Ascending  | 16 bytes| Byte 5    |

Note: The generated sort parameters are insufficient for stand-alone sort programs under z/VSE. If you want to use your own sort program, you must add 'WORK=' parameters to specify more than one file whose contents are to be sorted.

DBLUEREX User Exit

RELOAD modules IDMSDBL2, IDMSDBL3, and IDMSDBL4 have the ability to call a user exit named DBLUEREX when processing errors are encountered. This exit will be called if an unexpected error status is returned from a call to CA IDMS for all three modules. The exit is also called by IDMSDBL3 if an error is encountered while processing the intermediate file containing records to rebuild the database's chain sets.

The DBLUEREX module must be linked with each IDMSDBLx module from which the user wishes it to be invoked. The modules must be coded so that they are able to run with AMODE=31 and RMODE=ANY so that they will match the modes of the IDMSDBL? modules. The following statements should be used to link DBLUEREX with the desired module(s).

```
INCLUDE LIB1(DBLUEREX)
INCLUDE LIB2(IDMSDBL?)
ENTRY #DDNHD
```

where the ? represents 2, 3, or 4 depending on the module to which the exit is being linked.

Upon entry to DBLUEREX, Register 1 will point to a list of three full words with the following contents. The last full word in the list will have its high-order bit turned on to signal the end of the parameter list.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x'00'</td>
<td>Address of an eight byte field containing the name of the module making the call.</td>
</tr>
<tr>
<td>x'04'</td>
<td>Address of the error status field in the run-unit's subschema control for calls from IDMSDBL2, IDMSDBL4, and IDMSDBL3 if a database error is encountered. This parameter will contain zeros if called by IDMSDBL3 when an error is encountered while processing the intermediate file containing records defining the database's chain sets.</td>
</tr>
<tr>
<td>x'08'</td>
<td>Address of the current input record being processed by the calling module.</td>
</tr>
</tbody>
</table>

JCL Considerations

When you submit a RELOAD statement through the batch command facility, the JCL to execute the facility must include statements to define:

- Files containing the areas to be reloaded
Intermediate work files

Sort space

**Work file JCL considerations for STEP mode**

RELOAD normally runs as a single step but runs as separate steps using the "STEP step-name" syntax. When running in step mode, input files should have dispositions that state the file already exists, for example, "DISP=OLD".

Preserve output files on successful completion but not when the job fails, for example, "DISP=(NEW, CATLG,DELETE)."

See the "Intermediate Work File" table to determine which files are input and which files are output and when they are used.

The RELDCTL file is always input to every step.

The SYSPCH file is created by an IDMSDBLx step and used as input to a SORTx step. When used as input, it is defined as SYS001.

Work file record lengths:

- The RELDCTL file is a fixed-length file with a record length of 60 bytes.
- The SYSPCH file is a fixed-length file with a record length of 80 bytes.
- All SYSxxx files are variable length files. The record length can vary from one step to the next, from one job to the next. Do not code an LRECL value in the JCL, just code a BLKSIZE value. A BLKSIZE value should be chosen based on the optimal size for the device being used, for example, 1/2 track if disk or 32k if tape.

For more information about the generic JCL used to execute the batch command facility, see one of the following sample JCLs:

- Batch Command Facility (z/OS) (see page 355)
- z/VSE JCL (see page 389)

**Example**

The following example directs the RELOAD utility to reload a previously unloaded non-SQL-defined database.

```plaintext
reload;
```

The following command directs RELOAD to run all steps as a sortexit and to reuse workfiles:

```plaintext
reload as sortexit reuse workfiles;
```
Sample Output

The RELOAD utility generates the following listing after the successful execution of the RELOAD statement in the previous example. Note that the file the unloaded database is reloaded into is formatted first.

```
IDMSBCF mm/dd/yy PAGE 1

SET BATCH WIDTH PAGE 80;
Status = 0
FORM FILE USERDB.EMPF1;
File USERDB.EMPF1 blocks 1 to 50.
Area USERDB.EMP AREA pages 5,001 to 5,050.
Page size in file 4,096.

Status = 0
RELOAD ;
UT010002 BEGINNING PROCESSING FOR STEP SORT1
UT009001 IDMSDBLY RELEASE nn.n TAPE volser SORT STARTED
UT009002 105 RECORDS WERE READ FROM SYS002
UT009003 105 RECORDS WERE WRITTEN TO SYS004
UT009004 IDMSDBLY RELEASE nn.n SORT COMPLETED SUCCESSFULLY
UT010003 STEP SORT1 HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP IDMSDBL2
UT005001 IDMSDBL2 RELEASE nn.n TAPE volser PROCESSING STARTED
UT005002 DATABASE LOAD STATISTICS
DATABASE LOADED ON mm/dd/yy AT 123700
PAGES READ ................ 19
PAGES WRITTEN ............. 18
PAGES REQUESTED .......... 20
CALC RCDS IN TARGET PAGE . 12
CALC RCDS OVERFLOWED ...... 0
VIA RCDS IN TARGET PAGE .. 53
VIA RCDS OVERFLOWED ...... 1
LINES REQUESTED BY IDMS .. 222
RCDS MADE CURRENT OF R/U . 64
CALLS TO IDMS ............ 70
FRAGMENTS STORED .......... 0
RECORDS RELOCATED ........ 0
UT005004 77 INTERMEDIATE RECORDS WERE WRITTEN TO SYS006
UT005003 69 INTERMEDIATE RECORDS WERE WRITTEN TO SYS005
UT005004 SYS005 RECORD LENGTH IS 32
UT005005 NO DATABASE ERRORS WERE ENCOUNTERED
UT005006 IDMSDBL2 RELEASE nn.n PROCESSING COMPLETED
UT010003 STEP IDMSDBL2 HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP SORT2
UT009001 IDMSDBLY RELEASE nn.n TAPE volser SORT STARTED
UT009002 67 RECORDS WERE READ FROM SYS003
UT009002 77 RECORDS WERE READ FROM SYS006
UT009003 144 RECORDS WERE WRITTEN TO SYS007
UT009004 IDMSDBLY RELEASE nn.n SORT COMPLETED SUCCESSFULLY
UT010003 STEP SORT2 HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP IDMSDBLX
UT008001 IDMSDBLX RELEASE nn.n TAPE volser PROCESSING STARTED
UT008002 67 RECORDS WERE WRITTEN TO SYS008
UT008006 IDMSDBLX RELEASE nn.n PROCESSING COMPLETED
UT010003 STEP IDMSDBLX HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP SORT3
UT009001 IDMSDBLY RELEASE nn.n TAPE volser SORT STARTED
UT009002 69 RECORDS WERE READ FROM SYS005
UT009002 67 RECORDS WERE READ FROM SYS008
UT009003 136 RECORDS WERE WRITTEN TO SYS009
UT009004 IDMSDBLY RELEASE nn.n SORT COMPLETED SUCCESSFULLY
UT010003 STEP SORT3 HAS COMPLETED SUCCESSFULLY
UT010002 BEGINNING PROCESSING FOR STEP IDMSDBL3
UT006001 IDMSDBL3 RELEASE nn.n TAPE volser PROCESSING STARTED
```
The REORG utility reorganizes a database by unloading data from one or more areas in one database and reloading it into one or more areas of another database.

This article describes the following information:

- Authorization (see page 200)
- Syntax (see page 201)
- Parameters (see page 202)
- Usage (see page 205)
- JCL Considerations (see page 228)
- Examples (see page 230)
- Sample Output (see page 231)
- DBKEYS File Layout (see page 236)

### Authorization

<table>
<thead>
<tr>
<th>To</th>
<th>You Need This Privilege</th>
<th>On</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unload an area</td>
<td>DBAREAD</td>
<td>The area itself</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Any area(s) with set connections to the area</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Any area(s) containing an index defined on records in the area</td>
</tr>
<tr>
<td>Unload a segment</td>
<td>DBAREAD</td>
<td>All areas of the segment</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DBAWRITE</td>
<td></td>
</tr>
</tbody>
</table>
To | You Need This Privilege | On
---|---|---
Reload data into an area | The area itself Any area(s) with set connections to the area Any area(s) containing an index defined on records in the area |
Reload data into a database | DBAWRITE | All areas into which the data is being reloaded Any area(s) with set connections to target areas Any area(s) containing an index defined on records in the target areas

Syntax

```
REORG
  │setup-options[setup-execution-options]
  │     ESTIMATE workfile sizes
  │
  │execution-options
  │
  │CLEANUP
  │     NODELETE
  │     DELETEALL
  │     NOUNLOCK
  │
  │STATUS REPORT ONLY
  │   ESTIMATE workfile sizes
  │
  │SUBMIT

Expansion of setup-options

  │DBNAME source-db-name USING source-subschema-name
  │SEGMENT source-segment-name USING source-subschema-name

  │AREA area-name

  │RELOAD INTO target-database-name
  │USING target-subschema-name
  │DMCL target-dmcl-name

  │DIVIDE PROCESSING 1 n WAYS
  │REUSE workfiles
  │GENERATE DBKEYS files as SORTEXIT
  │SHARE
  │NOTIFY notify-interval
  │DELETE old workfiles
  │OVERFLOW PERCENT nnn
  │OVERFLOW CACHE nnnnnnn

Expansion of setup-execution-options

  │STOP AFTER CLEANUP SETUP UNLOAD RELOAD
  │CREATE NO UNLOAD RELOAD ALL WORKFILES

```
Expansion of execution-options

- **STOP AFTER**
- **CLEANUP**
- **UNLOAD**
- **RELOAD**
- **SUBMIT**
- **CREATE**
- **NO**
- **RELOAD**
- **ALL**
- **WORKFILES**

**Parameters**

- **CLEANUP**
  Specifies to perform a cleanup to delete work files and unlock target areas. You can use this option to complete a successful unload/reload operation if cleanup is not performed automatically. It can also be used to delete work files created by an incomplete unload/reload operation that is not being continued.

- **STATUS REPORT ONLY**
  Specifies to produce a status report by reading the reorganization control file. This can be run while other REORG jobs are executing to obtain a current status of the entire REORG process.

- **ESTIMATE workfile sizes**
  Directs REORG to estimate the size of work files by gathering statistics in a separate pass of the database. This option generates estimates for both UNLOAD and RELOAD work files. By default, statistics are collected during the unload phase that can be used for sizing RELOAD work files only. UNLOAD work files must be sized sectionally. If specified together with setup-options, statistics gathering starts as soon as the setup phase is complete and processing stops after the statistics have been gathered. Additional jobs are automatically submitted to help gather statistics and process the database by UNLOAD slice and index. If specified independently of setup-options, it must be specified in a separate execution of the REORG utility that occurs between the setup and UNLOAD phases. Multiple jobs are submitted to gather statistics only if the SUBMIT option is specified. Processing stops when file estimation is complete. When file estimation is complete, processing must be restarted by specifying a new STOP AFTER point. File size estimates are automatically used when dynamically allocating work files if no primary space value is specified for the file's DSMODEL.

- **source-db-name**
  Specifies the database name the source subschema with be bound to. All bound areas will be unloaded, unless restricted with the area-name option. The source subschema name must be specified with this option.

- **source-segment-name**
  Specifies the segment of the database whose areas are to be unloaded. The segment is referred to as the source segment, and the database is referred to as the source database.
- **source-subschema-name**
  Specifies the subschema that describes the source database.
  By default, if the source-subschema-name is not specified, it indicates that an SQL-defined segment is to be unloaded. In this case, the REORG utility extracts the description of the database from the dictionary to which the session is connected.

- **area-name**
  Specifies an area in the source segment to unload. Records in this area may target different areas in the target database.
  By default, if no areas are specified for a non-SQL defined database, all areas in the source segment that are also defined in the source subschema are unloaded. If no areas are specified for an SQL-defined database, all areas in the segment are unloaded.

- **Target-database-name**
  Specifies the name of the database into which data is reloaded. This is referred to as the target database.
  If an SQL-defined database is being reorganized, the **target-database-name** must be the same as the **source-segment-name** specified in the SEGMENT clause.
  If a non-SQL-defined database is being reorganized, **target-database-name** must be the name of either a segment or a DBNAME identifying the database into which data is to be reloaded.

- **Target-subschema-name**
  Specifies the subschema that describes the database into which the data is reloaded.
  Do not specify a target subschema if reorganizing an SQL-defined database.
  By default, if reorganizing a non-SQL-defined database and **target-subschema-name** is not specified, the source subschema is also the target subschema.

- **Target-DMCL-name**
  Specifies the DMCL that defines the database into which the data is reloaded.
  By default, if a **target-dmcl-name** is not specified, the DMCL that describes the source database is used during the reload phase and therefore must also describe the target database.
  See the source/target DMCLs in *Initiating a REORG Operation (see page 210)* for additional information.

- **DIVIDE PROCESSING n WAYS**
  Specifies the degree to which parallel processing is to be used to reduce elapsed time. The default is 1, which means that one job processes data and another processes system indexes.
  Use of this option exponentially increases the number of work files needed. For more information, see Work files and the n WAYS parameter.

- **REUSE workfiles**
  Specifies to reuse work files as much as possible. Use of this option reduces the number of work files that are needed.

- **GENERATE DBKEYS files**
  Specifies that each task that stores a record into the new database generates a DBKEYS file that maps the old db-key of the record with the new db-key of the record.
  For more information, see Generating db-key cross-reference information in *Using the REORG Utility (see page)*.
- **as SOREXIT**  
  Specifies to process sorted output data directly from the SORT, instead of writing it to an intermediate work file. Use of this option reduces the number of files and the amount of disk space needed.

- **SHARE**  
  Specifies to not lock source areas to prevent concurrent update. By default, source areas are locked at the start of unload processing.  
  When source areas are locked, they remain locked even after the REORG has completed. This prevents updates to the source area under the assumption that the target database is now the current database.

- **notify-interval**  
  Specifies a time-interval in minutes. Each time this interval expires, a message is written indicating the current task being processed and the percentage of records processed to that point. This message is not produced while data is being sorted.

- **DELETE old workfiles**  
  Directs REORG setup to delete work files that may be left from a previous run. All existing work files that match the name of a new work file are deleted, including DBKEYS files. For more information see, Considerations for running REORG on VSE.
  By default, old work files are reused.

- **OVERFLOW PERCENT nnn**  
  Specifies the percentage used to estimate the size of SYSOF2 and SYSOF8 work files and all output work files for the two overflow tasks: RELOAD2 and RELOAD5. The size of these files cannot be predicted, so they are estimated to be a percentage of related work files.
  By default, 10% is used.

- **OVERFLOW CACHE nnnn**  
  Specifies a limit on the size of the overflow cache used to temporarily hold records that do not fit on their target page.
  nnn is the maximum size of the overflow cache specified in bytes, Kilobytes (2**10), Megabytes (2**20), or Gigabytes (2**30), depending on whether it is followed by no qualifier or by KB, MB, or GB respectively.
  A value larger than 2**31-1 is limited to 2**31-1 bytes.
  By default, 32 KB is used.

- **STOP AFTER**  
  Specifies the processing phase after which execution halts. The options are:

  - **CLEANUP**  
    Continues processing until the entire reorganization operation is complete. CLEANUP is the default if setup-options are specified.

  - **SETUP**  
    Terminates processing after setup is complete and before unload is initiated. This option is valid only if setup-options are specified.

  - **UNLOAD**  
    Terminates processing after the unload phase is complete and before reload is initiated.
RELOAD
Terminates processing after the reload phase is complete and before cleanup is initiated.

If neither setup-options nor STOP AFTER is specified on a REORG statement, the processing stop point remains unchanged from that previously established. If STOP AFTER is specified, and the reorganization control file indicates that processing has already been initiated for a phase beyond that specified as a stop point, a warning is issued and processing is terminated.

SUBMIT
Specifies to submit JCL for additional REORG jobs to the internal reader when a REORG operation is restarted. This option is not valid during the initial setup phase, because by default, if processing continues beyond the initial setup phase, JCL is automatically submitted before starting the unload phase. The JCL to be submitted is retrieved from the RORGJCL file. If no RORGJCL is present in the execution JCL, the SUBMIT option is ignored.

CREATE NO/UNLOAD/RELOAD/ALL WORKFILES
Specifies whether to create work files and for which phases. The options are:

- NO
  Does not dynamically create work files. This is the default.

- UNLOAD
  Creates only the files used during unload processing.

- RELOAD
  Creates only the files used during reload processing.

- ALL
  Creates both unload and reload work files.

Note: For information about REORG’s work files and using data set models to create them, see Work Files (see page 216).

Usage
This section describes the reorganization process and provides section lines, and discusses considerations in the use of the REORG utility.

Submitting the REORG statement
You submit the REORG utility only through the batch command facility. You must run the batch command facility in local mode.

Processing Overview
The REORG utility unloads and reloads a CA IDMS database using parallel processing to reduce the amount of time it takes to reorganize data. To enable parallel operations, the utility divides the source and target databases into slices and groups of system-owned indexes. Each slice or index group can be processed (unloaded or reloaded) in parallel by concurrently executing jobs. The amount of concurrency is generally controlled by the DIVIDE PROCESSING n WAYS option; the higher the value specified, the more concurrency is possible.

A REORG operation is controlled through a control file that is built during operation initiation and is accessed by all jobs executing as part of the same REORG operation. The control file describes the specifics of the operation and is updated with status information reflecting the progress that has been made in operation execution. Each job that performs work on behalf of a REORG operation updates the control file. Operating system facilities are used to serialize these updates and to coordinate concurrently executing jobs.

A REORG operation is comprised of the following major execution phases:

- **Setup** -- in which the specifics of the operation are determined and its control file built.
- **Unload** -- in which the data is extracted from the source database and written to sequential files.
- **Reload** -- in which the extracted data is loaded into the target database and inter-record relationships and indexes are built.
- **Cleanup** -- in which work files may be deleted and target areas unlocked.

**General Reorganization Procedures**

Use the following procedures to reorganize a database. These procedures may vary slightly depending on the type of change being made and other operational considerations.

To REORG a database, take the following steps:

- Create the new subschema, segments and/or DMCL reflecting the changes to be made.
- Back up all areas that may be updated during the process.
  - Areas being unloaded that contain logically deleted records.
  - Areas being unloaded if data will be reloaded back into the same areas.
  - Areas physically linked to the unloaded areas through sets, linked constraints, or system indexes.
- If necessary, execute the CLEANUP utility to eliminate any logically deleted records from the areas being unloaded.
- If desired, execute the IDMSDBAN utility to ensure database integrity.

If the target database files are the same as the source database files, take the following steps:

- Execute the REORG utility specifying STOP AFTER UNLOAD.
If the SHARE option was not specified, use the UNLOCK command to unlock all areas into which the data will be reloaded.

- Format the areas into which the data will be reloaded.
- Execute the REORG utility specifying STOP AFTER CLEANUP and SUBMIT.
- Back up the updated areas. The new areas just loaded and any linked areas that were updated.
- If desired, execute the IDMSDBAN utility to ensure database integrity.

If the target database files are different from the source database files, and the two are described by different segment definitions whose files have different DDNAMEs and data set names, take the following steps:

- Format the areas into which the data will be reloaded.
- Execute the REORG utility specifying STOP AFTER CLEANUP.
- Back up the updated areas. The new areas just loaded and any linked areas that were updated.
- If desired, execute the IDMSDBAN utility to ensure database integrity.

If the target database files are different from the source database files, but the two have the same DDNAMEs, take the following steps:

- Format the areas into which the data will be reloaded.
- Execute the REORG utility specifying STOP AFTER UNLOAD. Ensure that the REORG JCL and the JCL in the RORGJCL file reference the source database files, the files containing the data to be unloaded.
- Execute the REORG utility specifying STOP AFTER CLEANUP and SUBMIT. Ensure that the REORG JCL and the JCL in the RORGJCL file reference the target database files, the files into which the data is to be loaded.
- Back up the updated areas. The new areas just loaded and any linked areas that were updated.
- If desired, execute the IDMSDBAN utility to ensure database integrity.

**Reducing REORG elapsed time**

You can substantially reduce the time it takes the REORG utility to unload and reload a database by using chained reads. This enables multiple pages to be read with a single I/O. Its use benefits both the unloading of data, which is primarily accomplished through an area sweep and the reloading of data, which REORG does by loading data in a forward direction. You will get the most benefit if you use chained reads for all areas being updated by the reload phase, which is most easily done using the PREFETCH facility. IDMSQSAM can be used instead, but because it applies to at most one area at a time, its benefit is limited.

**Relative REORG performance**
While in many cases REORG will require less time and resources to unload and reload a database than the UNLOAD/RELOAD utilities, this may not always be the case. For certain database structures, REORG will take longer to run and cost more in terms of CPU and I/Os. The best means of determining how REORG performs against a specific database is to do a trial run during a non-critical processing window. This should be done as part of planning a database reorganization.

Generating db-key cross-reference information

The REORG utility optionally creates a set of DBKEYS work files whose records relate the db-key of a record in the source database with its corresponding db-key in the target database. These files are not used by REORG, but can be used by a user-written application or a third-party product for the purpose of cross-referencing the old and new locations of a record. You may need to merge these files together prior to using the information.

> **Note:** For more information about and a description of the records in the DBKEYS files, see DBKEYS File Layout (see page 236).

Efficiency of the reorganized database

The database resulting from a REORG operation should be as efficient as one reorganized through UNLOAD and RELOAD even though the two may not be identical.

A database processed by RELOAD is loaded back to front. CALC and VIA records that overflow are usually written to pages that have already been loaded and have room. A database processed by REORG is loaded from front to back. CALC and VIA records that overflow are saved in a memory cache so that they do not displace records targeting later pages. If the cache is not large enough, the records are written to an overflow file and loaded in a later step.

The resulting databases should be similar in terms of the number of records that are stored on their intended target page and the number of records that overflow, but the two databases will not be exactly the same.

You can use the IDMSDBAN utility to obtain a report of the number of page changes needed to traverse all occurrences of each CALC and VIA set. By executing this utility before and after reorganization, you can determine the effect that REORG has had on these statistics and therefore the relative efficiency of the resulting database.

Implementing Changes Through REORG

Changing an SQL-defined database

You use REORG on an SQL-defined database to modify the database based on changes made to its segment definition. These changes must be reflected in the target DMCL. The modified DMCL must be available during the entire REORG operation, including the setup and unload phases.

You can make the following changes using REORG:

- Change area page ranges
Change page size
See EXPAND PAGE (see page 64) for an alternative means of changing a page size.

Change the maximum number of records per page
See CONVERT PAGE (see page 53) for an alternative means of changing the maximum number of records per page.

The names of the source and target segments for a REORG operation on an SQL-defined database must be the same. Consequently, if you are making any of the previous changes, you must specify a target DMCL whose name differs from that of the source DMCL. For more information about the DMCLs used by the REORG utility, see the source/target DMCLs in Initiating a REORG Operation (see page 210).

You cannot make changes to table definitions in an SQL-defined database between the setup and reload phases of a REORG operation.

Changing a non-SQL-defined database

You use REORG on a non-SQL-defined database to modify the database based on changes made to its schema or segment definitions. These changes must be reflected in the target subschema and DMCL. The modified DMCL and subschema must be available during the entire REORG operation, including the setup and unload phases.

You can make the following changes using REORG:

- Change area page ranges
- Change page size
  See EXPAND PAGE (see page 64) for an alternative means of changing a page size.
- Reassign records to different areas or page ranges

  Note: The record names must match in the source and target subschemas.

- Change record placements within an area
- Change record location modes
- Store VIA records by means of a different set (as long as they already participate in the set)
- Change compression and INDEX BLOCK CONTAINS options for indexes
- Change area and page-range assignments for system-owned indexes
- Change the maximum number of records per page
  See CONVERT PAGE (see page 53) for an alternative means of changing the maximum number of records per page.
- Add fields to an existing index key provided the fields already exist
- Remove fields from an index
- Change an index from unsorted to sorted provided the key fields already exist

  **Note:** Duplicate key order is random.

- Change an index from sorted to unsorted

**Changes that cannot be implemented using REORG**

The REORG utility cannot be used against native VSAM files.

During REORG processing, record formats and set relationships are preserved. This limits the modifications that you can make during a REORG operation. For example, you cannot use a REORG operation to:

- Remove a set
- Remove a record type
- Insert new data fields into a record
- Remove fields from a record
- Change the order of a sorted chained set
- Connect records to new chained or user-owned index sets
- Change the size, location, or data type of sort or index keys
- Change a chained set from unordered to ordered

These types of changes require other utilities, such as RESTRUCTURE, and possibly user-written programs.

**Initiating a REORG Operation**

**Specifying setup options**

A reorganization is initiated when a REORG statement is executed that specifies `setup-options`. The `setup-options` describe the segment to reorganize and the processing options to use. The first phase of the reorganization is called setup. During this phase, the parameters and the structure of both the source and target databases are analyzed to determine the tasks to perform and the work files that are needed. The result of this analysis is then written to the reorganization control file (RORGCTL). All subsequent reorganization jobs read the control file to determine what to do and how to do it.

**Important!** Be sure that setup-options are specified only when you intend to initiate a new REORG operation. Their presence causes the RORGCTL file to be overwritten, which results in the failure of any in-progress REORG operation using that control file.
The source/target subschemas

The REORG utility uses information in the dictionary when processing an SQL-defined database. When processing a non-SQL-defined database, the description of the data is contained in the source and target subschemas.

The source subschema describes the data to be unloaded and must:

- Include the areas being unloaded
- Include all areas containing records with set connections to records in the areas being unloaded
- Define all record types in the areas being unloaded
- Define all sets in which the record types participate
- Define all record types that can participate in the defined sets

The target subschema describes the data to be reloaded and must:

- Define the same record names and sets as the source subschema
- Include the areas into which data is to be reloaded and all areas containing records with set connections to those areas
- Allow all included areas to be readied in exclusive update mode

The source subschema must be available during the setup and unload phases. The target subschema must be available during the setup, unload, and reload phases.

The source segment and the target database

The source segment, which is the segment named in the SEGMENT clause of the REORG statement, must include all areas being reorganized. If these areas are physically connected to other areas, either:

- These other areas must be part of the source segment, or
- These areas must be part of segments that are included in a DBNAME whose name is the same as that of the source segment

The target database, which is the database named in the INTO clause of the REORG statement, must include all areas into which data is to be reloaded as part of the REORG operation. If these areas are in different segments or are physically connected to areas in other segments, the target database must be a DBNAME that includes all of these segments.

⚠️ **Note:** For more information about inter-area considerations, see Cross-area dependencies and Cross-segment dependencies in Special Database Considerations (see page 214).
The source/target DMCLs

The source DMCL must include the source segment and all segments with which the source areas are physically linked. The target DMCL must include the target segments and all segments with which the target areas are physically linked. Both the source and target DMCLs must be available during the setup and unload phases of the REORG operation. The target DMCL must also be available during the reload and cleanup phases.

The source DMCL is the DMCL specified in the SYSIDMS file. If no DMCL is specified in the SYSIDMS file, the DMCL named IDMSDMCL is used.

The target DMCL is the DMCL named in the INTO clause of the REORG statement. If no DMCL is specified in the INTO clause, the source DMCL is used as the target. If the same DMCL is used as both source and target, then either this DMCL must include both the source and target segments or no segment-related changes can be made to the data being reorganized.

Controlling REORG Execution

Specifying a stop point

By default, when a reorganization is initiated, it executes to completion or until a system failure or an unrecoverable error is encountered. However, you can break up execution into major processing phases by specifying a STOP AFTER parameter. You might want to halt processing after setup to examine the generated report to determine how much disk space is needed. Or, you might halt execution after the unload phase to delete the source database files before allocating the target files. Once a stop point has been established, which always occurs during setup, it remains in effect until another REORG statement is executed that specifies a different STOP AFTER value. The specified stop point must be the same as or later than the reorganization phase most recently processed.

Automatic job submission

The REORG utility relies on concurrently executing jobs for its parallel processing. To facilitate this, REORG has the ability to automatically submit jobs through the internal reader. It does this at the end of the setup phase unless STOP AFTER SETUP has been specified. It also does this anytime a REORG statement is executed that specifies SUBMIT. Typically, the SUBMIT keyword is used only when restarting a reorganization and no other jobs are active, such as when execution has been interrupted because a stop point has been reached or an error such as a system failure has caused all jobs to fail. If specified at other times, it may result in more jobs than there is work to perform, therefore needlessly tying up initiators.

JCL for jobs submitted automatically is retrieved from the RORGJCL file. The JCL should have the following characteristics:

- The prefix of the included job name should be short enough to allow for appending one or more characters to make the job name unique.
- The JCL should include a step that executes a REORG statement through the batch command facility. No parameters should be specified on the REORG statement.
- The RORGJCL file should not be included.
- The JCL should have the same considerations as listed in JCL Considerations (see page 228).
The JCL or the DMCL(s) being used should either allow simultaneous access to both the source and target files or you must stop execution between the unload and reload phases and may need to change the RORGJCL contents so that instead of referencing the source files, it references the target files. Failure to do this can result in corruption of the source database as REORG may attempt to reload into the source files.

Note: For more information about the JCL needed to execute a REORG statement, see the JCL Considerations (see page 228).

REORG submits one job for each slice and each index group up to two times the number of slices. If there are two slices and one index group, three jobs are submitted. If there are two slices and three index groups, four jobs are submitted.

**Manual job submission**

Once a REORG operation has been initiated, you can sectionly submit as many REORG jobs as you want. Each job examines the control file to determine if there is any work to do on behalf of the reorganization. If no such work exists, perhaps because pre-requisite tasks have not yet completed, the job waits until some other job completes a task before re-examining the control file. If it finds work to do, it begins to perform that work provided no other job claims it first.

While section job submission is possible, automatic submission should normally be used to avoid having either too few or too many jobs active at one time. Manual job submission can be used to resubmit one or more failed jobs.

**Restarting REORG**

If a REORG job fails, other REORG jobs try to automatically restart the task that failed after they have completed the task they were processing. If the problem was a temporary environment-related problem, and the task successfully completes, nothing additional needs to be done. Although, you may want to submit another REORG job to make up for the one that failed.

If the problem was not temporary, for example, a work file was too small; the remaining REORG jobs also fail when they try to restart the task, and at some point all jobs end. In this case, if the problem can be corrected, REORG can be restarted simply by resubmitting one or more REORG jobs. If the first one specifies the SUBMIT option, it submits the additional jobs.

REORG automatically restarts at the failing task or in some cases on a prior task if that is what is needed to recover from the failure. Depending on the requirements of the failing task, a portion of the database may be automatically formatted, or an index may be deleted as part of the restart process.

**Restarting REORG from the beginning**

Some problems may be severe enough to require restarting REORG from the beginning, such as if the wrong subschema was specified causing REORG to fail because the data did not match its description. Should it be necessary to restart REORG processing from the beginning, take the following actions:
Format the target database if it had been partially loaded

- Delete any automatically created work files by executing a REORG statement that specifies CLEANUP

- Restart the REORG operation by executing a REORG statement that specifies setup-options, corrected if necessary.

Recovering from an out-of-space condition on a work file

You can recover from an out-of-space condition on a work file by taking the following actions:

- Manually delete the work file and reallocate it with more space

- Resubmit one or more REORG jobs. If all previously active jobs failed, submit only one job and specify the SUBMIT option to enable the automatic submission of the remainder of the jobs. If some jobs are still active, you can sectionly submit a job for each one that failed.

REORG serialization in a multi-image environment

Serialization is used to coordinate execution between jobs concurrently executing a REORG operation. These jobs serialize access to various internal resources by using a SYSTEMS ENQ with a QNAME of IDMSUT. In a multi-image environment, the SYSTEMS ENQ allows REORG jobs running on different images to serialize properly. However, if a multi-image manager is used, a SYSTEMS ENQ may be converted to a SYSTEM ENQ, which would only serialize REORG jobs running on the same image. In this environment, either restrict REORG jobs to run on one image or configure the multi-image manager to exempt REORG ENQ requests from being converted.

Special Database Considerations

Reorganizing a dictionary

When reorganizing a dictionary, the source and target subschemas are always the same and are determined by the areas of the dictionary being reorganized.

- Use the IDMSNWKU subschema for all areas, except DDLCAT, DDLCATX, and DDLCATLOD.

- Use the IDMSCATZ subschema for the DDLCAT and DDLCATX areas.

- Use the IDMSCATL subschema for the DDLCATLOD area.

Before reloading the DDLCATLOD area in a segment defined for SQL, you must install stamps either by formatting by area or using the INSTALL STAMPS utility.

REORG an ASF database

If you want to run the REORG utility against an ASF data or definition area, see the CA IDMS ASF User Section for more information. The CA IDMS ASF User Section refers specifically to the UNLOAD and RELOAD utilities for unloading and reloading an ASF database, but REORG may be substituted for UNLOAD/RELOAD in this case.

Logically deleted records
You must remove any logically deleted records from the areas being unloaded before executing the REORG utility. To determine if an area has logically deleted records, use PRINT SPACE with the FULL option. If logically deleted records exist, use the CLEANUP utility to remove them from the area.

**SQL-defined databases and synchronization stamps**

Area and table synchronization stamps in SQL-defined segments are unloaded and reloaded by the REORG utility. For this reason, you must format the target areas using the FILE option so that the format operation does not also store synchronization stamps.

The one exception to this is when unloading and reloading the DDLCATLOD area in a segment defined for SQL. In this case, use the AREA option of the FORMAT statement or use the INSTALL STAMPS utility so that the format operation stores synchronization stamps.

**Cross area dependencies**

Areas that have physical connections to the areas being reorganized are referred to as **dependent areas**. You can reorganize an area that has physical connections to other areas without also unloading and reloading its dependent areas. Physical connections include:

- Sets that cross area boundaries
- Indexes that cross area boundaries
- Linked constraints in which the referenced and referencing tables reside in different areas

The REORG utility keeps track of the inter-area linkages and rebuilds them during the RELOAD phase. For this reason, not only must the target areas be updatable by REORG, but their dependent areas must also be updatable.

**Cross segment dependencies**

A source area can have physical connections to an area in another segment. To reorganize all related areas, the DBNAME setup option must be used, and it must include all segments with physical connections. When the SEGMENT setup option is used, only areas in the name segment can be reorganized. Areas in connected segments will be processed as dependent areas. Their pointers will be adjusted, but they will not be reorganized. To use the SEGMENT option, the segment named on the setup clause of the REORG statement must also be the name of a DBNAME that includes both the source segment and all segments whose areas are physically connected to the areas being reorganized.

A target area can have physical connections to an area in another segment. To reorganize into such an area, the INTO clause of the REORG statement must identify a DBNAME whose segments include all of the target areas and all of their dependent areas.

**Mixed page groups**

Mixed page groups are supported. A source DBNAME may include segments with different page-group and different maximum records-per-page values. A segment may have physical connections to segments with a different page-group and/or maximum records-per-page value. Cross segment chained sets must have the same page-group and maximum records-per-page.

**Ignoring the order of CALC duplicates**
By default, when REORG unloads a CALC record for which duplicates are allowed, it preserves the order of duplicate entries by walking the CALC set in a prior direction and determining the relative position of the record with respect to others with the same CALC key value.

This processing can result in significant overhead. If preserving the order of duplicate entries is not important, you can eliminate this processing by turning on the OPT00093 bit in the RHDCOPTF module.

**Processing of DIRECT records**

Because DIRECT records have been placed on a specific page by a user, the REORG utility may have difficulty in determining where they should be placed in the target area. If a DIRECT record's old page exists in its target page range, REORG attempts to place the occurrence on the same page. However, if the old page does not exist in the target page range, the record is stored in the target area proportionally to its position in the source area.

If this is not acceptable, you may need to write a user-written program to unload the database and the FASTLOAD utility to reload it.

**DCMT VARY PERMANENT considerations**

If you run the REORG utility to change an area's low page number and Change Tracking is not used, it is recommended that you remove any permanent status on the affected area before making the new DMCL active within the CV.

When Change Tracking is not used, the PERMANENT feature is implemented by carrying the area's low page number in the journals across cycles of the CV. Changing an area's low-page prohibits future cycles of the CV from properly identifying the area once the new page range is implemented.

If a DCMT VARY SEGMENT/AREA PERMANENT command is still in effect when the new page range is implemented, the area's usage-mode at startup is determined by the value specified in the DMCL. The entry in the journals for the old area's page range remains until the next format of the journals.

The journal entry for the old starting page can be removed without formatting the journal by doing a non-permanent DCMT VARY AREA command against the area prior to changing the DMCL definition in the CV.

**Work Files**

The REORG process requires many work files. These files can be dynamically created and allocated for you by the REORG utility, or you can sectionly assume these responsibilities. This section discusses considerations associated with the creation and use of work files.

**Work files and the n WAYS parameter**

The number of work files required by REORG can grow exponentially as the value in the DIVIDE PROCESSING n WAYS parameter is increased. Therefore, you should exercise care in choosing this parameter.

For example, a REORG that specifies DIVIDE PROCESSING 2 WAYS usually results in 2 unload slices, 2 reload slices, and 4 SYS002 work files. If 3 WAYS were specified, this usually results in 3 unload slices, 3 reload slices, and 9 SYS002 work files. A similar expansion exists for each RELOAD phase.
REBUILD phases use fewer work files, but the number of index work files generated by RELOAD1 increases with the number of reload slices.

**Reducing the number of work files**

There are two ways to reduce the number of work files needed by the REORG utility other than reducing the n-WAYS parameter. You can direct REORG to:

- Reuse work files, which allows work files that are no longer needed to be reused for other purposes during later phases of processing
- Execute as a sort exit, which eliminates the use of sorted output files

It is recommended that both of these options be used to reduce the number of work files, and the amount of disk space needed.

**Work file creation**

REORG can create work files dynamically or you can sectionly create them prior to beginning the unload and/or reload phases of REORG execution. Regardless of how the files are created, it is a good idea to halt execution after setup to determine what work files are needed by examining the report produced by REORG.

If using dynamic work file creation, you must specify the attributes of the work files using one or more CREATE DSMODEL statements. REORG creates the files either as directed by the CREATE WORKFILE clause or at the time they are first accessed. Dynamically created work files, other than DBKEYS files, are deleted automatically during the cleanup phase.

If you want to use REORG’s size estimates to create a file, code a DSMODEL without a primary SPACE allocation. You can code a SPACE parameter with just a unit type (TRK, CYL, or blksize) and no value for primary allocation.

You must code a primary space allocation value or delay creating work files until estimates are available. This means, for example, that you cannot direct REORG to create RELOAD work files during the setup phase unless the DSMODEL contains a primary allocation value.

If you code a zero primary space allocation value and a non-zero secondary value, the secondary value is replaced by a value derived from the estimated primary value.

**Note:** For more information, see Considerations for running REORG on VSE.

**Work file allocation**

REORG can allocate work files dynamically or you can reference them sectionly by including an appropriate DD statement in the REORG job’s JCL and in the JCL included in the RORGJCL file if automatic job submission is used.

If using dynamic work file allocation, you must specify the data set names of the work files using one or more CREATE DSMODEL statements. REORG allocates the files as they are needed. If a DD card for a given work file is included in the execution JCL, it is used, and that file is not dynamically allocated.
If not using dynamic work file allocation, you must include a DD statement for every work file in every REORG job and in the RORGJCL file.

**Work file deletion**

By default during the cleanup phase, REORG deletes all work files created during the current operation other than DBKEYS files. It deletes only those files for which a matching data set model was specified either at setup or when the file was created. If a matching model is detected, REORG attempts to dynamically allocate the file thereby determining its data set name which may be derived from the model or overridden by a DD statement in the JCL. Regardless of how the data set name is determined, if the file is created during the execution of a REORG statement, it will be deleted during the cleanup phase of the operation unless the file is subsequently overridden with a different data set name in some later job. You can determine which files cleanup will delete by examining the work file summary sections of the REORG Status Report.

Should it be necessary to restart a REORG operation from the beginning, you should first execute a REORG statement that specifies CLEANUP to delete any work files created by the interrupted operation. If you do not do this, none of the work files created during the first operation execution will be deleted by REORG even if they are reused during the second execution.

If REORG is restarted without first cleaning up the old work files, you can still direct REORG to delete them using one of the following methods:

- You can allow REORG to reuse the old files and then after REORG has ended normally, run a REORG CLEANUP job with the DELETEALL option. This option directs REORG to delete all work files, other than DBKEYS files, whether they were created by the most recent REORG operation or not.

- You can delete the old work files during the setup phase of the restarted run. Specify DELETE OLD WORKFILES during setup and REORG checks for and deletes any files it finds that match the file names it plans to use. This includes DBKEYS files. This approach is preferable when the old files may be too small to be reused.

**Note:** For more information, see Considerations for running REORG on VSE.

**Using DSMODELs with REORG**

Data set models (DSMODELs) are used to specify a set of attributes for data sets. REORG can use DSMODELs to specify the attributes for its work files, thereby eliminating the need for coding DD statements for every work file used by the utility.

The REORG utility checks for the presence of DSMODELs during the setup phase. If they are present, REORG saves the model information in the control file for later use. This saved information is used by subsequent tasks and by other jobs. No additional DSMODEL statements need to be specified in any submitted or subsequent job.
When REORG creates a work file, either because a CREATE WORKFILE clause has been specified or because a file does not exist at the time it is accessed, REORG uses the saved model information unless another CREATE DSMODEL statement has been specified in the same session as the one in which the file is being created. If another CREATE DSMODEL statement has been specified, it overrides the previously saved information. REORG updates the control file with the new information and uses it to create the work file.

To use a data set model's attributes for a given work file, the model's name must match the file's DDNAME according to the rules discussed in the CREATE DSMODEL statement. The REORG utility uses the following DDNAME prefixes:

- WU -- Unload/Reload work files
- WI -- Index REBUILD work files
- WS -- Sort output work files
- WD -- DBKEYS work files

Each prefix is followed by a generated number. To use a model for a work file, the model name must match the file's DDNAME. If more than one DSMODEL matches that of a work file, the one that most closely matches is used. For information about how to specify model names that match character strings such as a DDNAME, see CREATE DSMODEL (see page 61).

For the REORG utility to dynamically allocate a work file, its matching model must minimally specify a DSN attribute so that the name of the data set can be determined. Symbolic parameters in the DSN value can be used to generate different names for each work file.

For the REORG utility to create a work file, the DSMODEL must also include information to identify where to create the file, the space to allocate, and a block size to use.

When specifying a DSMODEL for a DBKEYS file, the BLKSIZE must be a multiple of 16. All other work files are variable length. The BLKSIZE for these files must be four bytes larger than the largest work record.

If a block size is not coded or BLKSIZE 0 is coded in the DSMODEL for a REORG file, REORG chooses one based on the device type. If the device type cannot be determined, a 3390 will be assumed. Normally, 1/2 track blocking is used for a 3390 device.

Sizing work files

The simplest way to size work files is to let REORG do it for you. REORG automatically estimates the size of all files after making a pass of the database and gathering statistics. This occurs by default when the data is unloaded. While this does not require an extra pass of the data, the estimates that are generated can only be used for allocating RELOAD work files because the UNLOAD files have already been created and used.

To use REORG-generated estimates for allocating UNLOAD work files, use the ESTIMATE WORKFILE SIZES option. This directs REORG to make a preliminary pass of the data without opening or writing to any work files. The generated file estimates are stored in the control file and can be used to allocate both UNLOAD and RELOAD work files when the REORG operation is resumed.
For REORG to more accurately estimate the size of overflow files, it may be necessary to specify an OVERFLOW PERCENT parameter. REORG assumes that 10% of the records to be stored in the database will overflow. If this assumption is not valid for a particular database, you may need to specify a different overflow percent value so that REORG can generate better estimates for the size of overflow files.

While REORG attempts to accurately estimate the size needed for work files, it may not always be able to do so. In certain cases, it may be necessary to sectionly estimate the size needed for one or more work files.

Estimating the size needed for work files is difficult for two reasons: many classes of files contain different types of records, and the number of records written to each file within a class varies depending on how REORG chooses to divide the page ranges into slices. Consequently, there are no simple formulas that can be used to estimate work file sizes. There are however, some techniques that can be used to facilitate file sizing and allocation.

In planning for a reorganization, do one or more trial runs to determine the actual file sizes needed for a given n-WAY value. This can be done during a non-critical time against a copy of the source database. This is the easiest way to obtain accurate size estimates. If it is impractical to do a trial reorganization on a full copy of the database, do it on a reasonably-sized sample that is representative of the original and then scale up the work file sizes proportionately. Be sure that the sample database is large enough relative to the n-WAY parameter so that the slicing algorithm does not reduce the number of slices due to their small size. Size estimates determined using a sample database will not be as accurate as those determined using a full copy of the database and so should be increased to account for this.

The following is a list of additional techniques that may prove helpful in allocating work files under z/OS:

- Specify a relatively modest primary allocation and a large secondary allocation. This allows the file to extend to handle large amounts of information without wasting unneeded space.

- Use IBM’s Storage Management Subsystem (SMS) to determine where files are to be allocated rather than trying to sectionly place files on specific volumes. SMS can automatically spread the files across volumes wherever there is available space. Use the DATACLAS and STORCLAS options to allow files to extend across multiple volumes.

- If SMS is not available, allow the files to be allocated across multiple volumes by specifying multiple volume serial numbers. There must be enough space across all listed volumes to satisfy the space needs of all of the work files. The order of the entries in the list is not important.

- Consider using extended format data sets for large work files. Although this adds 32 bytes of overhead to each block, extended format data sets can have more extents across more volumes than a basic format data set.

Under z/VSE it is recommended to not specify a primary space value, and let REORG calculate one. If this value is not large enough, or will not fit on the specified volume, a DLBL and EXTENT for the individual file should be sectionly coded to override the generated label.

**Work file restrictions**

You cannot use tape or virtual tape for work files.
REORG Processing Details

How the unload phase works

During the unload phase, most record types are unloaded as part of an area sweep. Certain database structures necessitate exceptions to this general processing flow.

- A record that is stored VIA a System Index in which sequence order cannot be determined from the member record (such as an unsorted index or one in which duplicates are FIRST or LAST), is offloaded by walking its index.

- Members of a VIA set are unloaded at the time their owner is unloaded by walking the VIA set. If the VIA member in turn owns VIA members, they too are unloaded in a similar fashion.

- When unloading a CALC record for which duplicates are allowed, the CALC set is walked in the prior direction to determine the relative position of the record with respect to its duplicates.

Records that are unloaded are assigned a target page number during the unload phase. Target page numbers are assigned as follows:

- CALC records are assigned a target page based on their CALC key and target page range

- VIA records that are members of their VIA set are assigned a target page based on their owner’s assigned or target page. If a VIA record is in the same target page range as its owner, it is assigned the same target page plus any VIA displacement. If in a different page range, it is assigned a target page in its range proportional to its owner’s page in its assigned or target page range plus any VIA displacement.

- VIA records that are not members of their VIA set are assigned a target page that is proportional to their position in their source page range.

- DIRECT records are assigned a target page that is the same as their source page if the page is in both the source and target page ranges; otherwise, DIRECT records are assigned a target page that is proportional to their position in their source page range.

REORG slicing

A database is divided into slices based upon the value coded in the DIVIDE PROCESSING n WAYS parameter. Generally, each area is divided symmetrically, but if an area contains a subarea and no outside records span that subarea, REORG divides the subarea independently of the rest of the area.

If dividing an area or subarea produces a piece that is too small, smaller than 100 pages, that area or subarea is divided again into fewer, but larger pieces. If an area or subarea is less than 200 pages, it is not divided at all.

After all areas and subareas have been divided into pieces, each piece is assigned to a slice. Each slice typically ends up with approximately the same number of pages.

Note: For more information about the types of work files used by the REORG utility, see the Classes of work files in REORG Processing Details (see page 221).
Usually, the number of slices equals the n-WAYS value, but if the total number of pages being processed is small, there may be fewer slices. If the total size of all areas is less than 200 pages, all pages are assigned to one slice, regardless of the n-WAYS value.

Dependent areas are areas with physical linkages to the areas being processed. They are also divided and assigned to slices.

Areas and subareas that contain only system indexes are not divided and are not assigned to a slice.

The source and target database are sliced independently of one another and consequently may not be divided into the same number of slices.

**Index groups**

A user-owned index occurrence is processed with the slice to which its owner record is assigned. System-owned indexes are assigned to groups and processed independently of slices. System indexes that have the same or overlapping target page ranges are assigned to the same group; otherwise, they are assigned to different groups. If an index has a distinct (non-overlapping) target page range, it is assigned its own index group. Index groups are processed independently of each other and therefore can be processed in parallel. Once assigned to a group, an index is in that group for the duration of REORG processing.

**REORG tasks and phases**

REORG processing is divided into tasks and grouped into phases. Each phase processes a type of work needed to unload or reload a database or rebuild an index. Each task processes a slice or index group within a phase.

For example, if a database were divided into two unload slices and had three index groups, the unload phase could have up to five tasks: one for each slice and possibly one for each index group. All unload tasks must successfully complete before REORG can begin to process tasks in the RELOAD or REBUILD phases.

RELOAD contains six phases numbered 1 through 6. These phases reload slices, rebuild user indexes, and reconnect pointers. Each reload task in a given phase must successfully complete before processing can move to the next RELOAD phase.

REBUILD can contain up to three phases numbered 1 through 3. Each index group has one task per phase, but each index group can run independently of the others and independently of RELOAD phases 3 through 5.

If a system index related page range conflict exists between tasks in RELOAD and REBUILD, or between REBUILD tasks, updates of the page range are serialized.

The phases are:

<table>
<thead>
<tr>
<th>Phase Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SETU</td>
</tr>
<tr>
<td>UNLO</td>
</tr>
<tr>
<td>AD</td>
</tr>
<tr>
<td>Phase Description</td>
</tr>
<tr>
<td>-------------------</td>
</tr>
<tr>
<td><strong>Consists of Unload Slice and Unload Index tasks.</strong> Slices are offloaded through area sweeps and by walking VIA sets. If an index needs its sequence preserved, the index is walked to determine the relative sequence number of each member. The sequence data is merged with information extracted from the loaded record in REBUILD2. If VIA System Indexes exist and their sequence needs to be preserved, the member records are offloaded by walking the index. VIA members are offloaded when their owners are offloaded regardless of how the owners are offloaded.</td>
</tr>
</tbody>
</table>

| REBU This phase exists only when there is a VIA SYSTEM INDEX that spans multiple reload slices. If needed, it runs prior to RELOAD1. It sorts VIA INDEX member records into index order and assigns the records to target slices. | |

| RELO Loads data into the target database. Set connection and index key data is generated and processed by later phases. If a task in this phase fails, when restarted, it formats only the page ranges assigned to its slice and restarts processing from the beginning. | |

| RELO This is an overflow phase. If a reload slice fills up, the data is saved and processed by this phase. There is only one task in this phase, and it updates across slices. It generates the same set connection and index key data as RELOAD1 for records it stores. If an area fills up during this phase, processing stops. If the task in this phase fails, when restarted, REORG starts over by reprocessing all RELOAD1 tasks. Each formats its slice and restarts. | |

| RELO Merges user-owned index data generated by UNLOAD, RELOAD1, and RELOAD2. Member keys and/or sequence numbers are merged in owner sequence and generated records are passed to RELOAD4. | |

| RELO Merges data produced by RELOAD1 and RELOAD2 about old and new db-keys for chained sets and passes it to RELOAD6. If data is passed from RELOAD3, it rebuilds user indexes using that data. INDEX and OWNER pointer data is generated and passed to RELOAD6. If a task in this phase fails, when restarted, it reprocesses data from the beginning. It erases and rebuilds all user index occurrences in its slice. If a target slice fills before all data is processed, unconnected data is passed to RELOAD5. Tasks in this phase serialize on system index related page ranges. | |

| RELO This is an overflow phase for RELOAD4. Data that would not fit into a slice is processed here. There is only one task in this phase, and it updates pages across all slices. If an area fills up during this phase, processing stops. Index key connection begins with the key that failed in RELOAD4. If the task in this phase fails, when restarted, REORG reprocesses all tasks in RELOAD4. User indexes that were already built are rebuilt. Tasks in this phase serialize on system index related page ranges. | |

| RELO Updates prefix pointers using data generated by RELOAD phases 4 and 5; and REBUILD phase 3. | |

| RELO Updates prefix pointers using data generated by RELOAD phases 4 and 5; and REBUILD phase 3. | |

This phase processes those records that were relocated out of their target slice and could not be processed during a RELOAD6 phase. There is only RELOAD7 phase which processes across slice boundaries.
### Phase Description

**REBU**  This phase merges sequence data from an **UNLOAD INDEX** task with member data from **RELOAD1** and **RELOAD2**.

It is only generated for groups that have indexes for which sequence must be preserved.

It can run in parallel with all **REBUILD** phases 2 through 4 for other index groups and in parallel with **RELOAD** phases 3 through 6.

**ILD2**  This phase uses member data from **RELOAD1**, **RELOAD2**, and **REBUILD2** to rebuild system indexes.

Tasks in one group can run in parallel with reload tasks and index tasks from other groups as long as there are no page range conflicts.

Tasks in this phase serialize on system index related page ranges.

**CLEAN**  There are no tasks in this phase. When all other processing has successfully completed, this phase optionally unlocks the reload areas and deletes work files created by the **REORG** utility.

### Classes of work files

Each phase of **REORG** processing can produce different classes of work files. A task can produce one or more occurrences of a given class of work file. Each work file is read by a task in a later phase along with similar work files from other tasks.

The following table describes each class of work file. When discussing how many work files are generated, sometimes a task’s phase name is referenced and sometimes the number of reload or unload slices is referenced. For each unload slice, there is one **UNLOAD SLICE** task and one **RELOAD3** task. For each reload slice, there is one **RELOAD1**, **RELOAD4**, and **RELOAD6** task. The number of reload slices is the same as the number of **RELOAD1** tasks. Sometimes these terms are interchanged when it is thought to clarify a description.

The classes of work files are:

<table>
<thead>
<tr>
<th>Work File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS002</td>
<td>Contains offloaded record and set information.</td>
</tr>
<tr>
<td></td>
<td>Output: <strong>UNLOAD SLICE</strong> and <strong>UNLOAD INDEX</strong></td>
</tr>
<tr>
<td></td>
<td>Input: <strong>RELOAD1</strong></td>
</tr>
<tr>
<td></td>
<td>One is created for each unload slice times the number of reload slices.</td>
</tr>
<tr>
<td></td>
<td>One is created for each index group that contains a VIA system index where all members target one reload slice.</td>
</tr>
<tr>
<td>SYSX02</td>
<td>Contains offloaded record and set information for members of a VIA index.</td>
</tr>
<tr>
<td></td>
<td>Output: <strong>UNLOAD SLICE</strong> and <strong>UNLOAD INDEX</strong></td>
</tr>
<tr>
<td></td>
<td>Input: <strong>REBUILD1</strong></td>
</tr>
<tr>
<td></td>
<td>One is created for each unload slice times the number of index groups that contain VIA system indexes that target multiple reload slices.</td>
</tr>
<tr>
<td></td>
<td>One is created for each index group that contains VIA system indexes that target multiple reload slices.</td>
</tr>
<tr>
<td>SYSIX2</td>
<td>Contains sorted record and set information for members of a VIA index.</td>
</tr>
<tr>
<td></td>
<td>Output: <strong>REBUILD1</strong></td>
</tr>
<tr>
<td></td>
<td>Input: <strong>RELOAD1</strong></td>
</tr>
<tr>
<td></td>
<td>One is created for each index group that contains VIA system indexes that target multiple reload slices times the number of reload slices.</td>
</tr>
<tr>
<td>Work File Classes</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>SYSOF 2</td>
<td>Contains record and set information for records that overflowed their target slices.</td>
</tr>
<tr>
<td></td>
<td>Output: RELOAD1</td>
</tr>
<tr>
<td></td>
<td>Input: RELOAD2</td>
</tr>
<tr>
<td></td>
<td>One is created for each reload slice or RELOAD1 task.</td>
</tr>
<tr>
<td>SYS003</td>
<td>Contains offloaded index sequence information for user indexes.</td>
</tr>
<tr>
<td></td>
<td>Output: UNLOAD SLICE</td>
</tr>
<tr>
<td></td>
<td>Input: RELOAD3</td>
</tr>
<tr>
<td></td>
<td>One is created for each unload slice.</td>
</tr>
<tr>
<td>SYSX03</td>
<td>Contains offloaded index sequence information for system indexes.</td>
</tr>
<tr>
<td></td>
<td>Output: UNLOAD INDEX</td>
</tr>
<tr>
<td></td>
<td>Input: REBUILD2</td>
</tr>
<tr>
<td></td>
<td>One is created for each index group that contains system indexes that are not VIA and where member sequence is not being preserved.</td>
</tr>
<tr>
<td>SYS004</td>
<td>Work file containing sorted record and set information.</td>
</tr>
<tr>
<td></td>
<td>Not created when SORTEXIT is specified.</td>
</tr>
<tr>
<td></td>
<td>One is created for each RELOAD1 and RELOAD2 task.</td>
</tr>
<tr>
<td>SYS005</td>
<td>Contains old and new db-key values for chained sets.</td>
</tr>
<tr>
<td></td>
<td>Output: RELOAD1 and RELOAD2</td>
</tr>
<tr>
<td></td>
<td>Input: RELOAD4</td>
</tr>
<tr>
<td></td>
<td>One is created for each (RELOAD1 and RELOAD2 task) times the number of RELOAD4 tasks.</td>
</tr>
<tr>
<td>SYS006</td>
<td>Contains user index owner and member db-key and symbolic key information.</td>
</tr>
<tr>
<td></td>
<td>Output: RELOAD1 and RELOAD2</td>
</tr>
<tr>
<td></td>
<td>Input: RELOAD3</td>
</tr>
<tr>
<td></td>
<td>One is created for each reload slice times the number of unload slices.</td>
</tr>
<tr>
<td>SYSX06</td>
<td>Contains system index member db-key and symbolic key information.</td>
</tr>
<tr>
<td></td>
<td>Output: RELOAD1 and RELOAD6</td>
</tr>
<tr>
<td></td>
<td>Input: REBUILD2</td>
</tr>
<tr>
<td></td>
<td>One is created for each reload slice times the number of index groups that contained a non-VIA system index where sequence was being preserved.</td>
</tr>
<tr>
<td>SYS007</td>
<td>Work file containing sorted user index owner and member work record information.</td>
</tr>
<tr>
<td></td>
<td>Not created when SORTEXIT is specified.</td>
</tr>
<tr>
<td></td>
<td>One is created for each RELOAD3 task.</td>
</tr>
<tr>
<td>SYSX07</td>
<td>Work file containing sorted user index owner and member work record information.</td>
</tr>
<tr>
<td></td>
<td>Not created when SORTEXIT is specified.</td>
</tr>
<tr>
<td></td>
<td>One is created for each REBUILD2 task.</td>
</tr>
<tr>
<td>SYS008</td>
<td>Contains user index information that is used to rebuild indexes.</td>
</tr>
<tr>
<td></td>
<td>Output: RELOAD3</td>
</tr>
<tr>
<td></td>
<td>Input: RELOAD4</td>
</tr>
<tr>
<td></td>
<td>One is created for each unload slice times the number of reload slices.</td>
</tr>
<tr>
<td>SYSX08</td>
<td>Contains system index information that is used to rebuild indexes.</td>
</tr>
<tr>
<td></td>
<td>Output: RELOAD1, RELOAD2, and REBUILD2</td>
</tr>
<tr>
<td></td>
<td>Input: REBUILD3</td>
</tr>
<tr>
<td></td>
<td>One is created for each REBUILD2 task that is generated. Also, one is created for each index group (that contains a VIA system index or a non-VIA system index where sequence was not being preserved) times the number of RELOAD1 and RELOAD2 tasks.</td>
</tr>
</tbody>
</table>
### Work File Classes

<table>
<thead>
<tr>
<th>Work File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS009</td>
<td>Work file containing sorted user index work records and sorted old/new db-keys for chained sets. Not created when SORTEXIT is specified. One is created for each RELOAD4 and RELOAD5 task.</td>
</tr>
<tr>
<td>SYSX09</td>
<td>Work file containing sorted system index work records. Not created when SORTEXIT is specified. One is created for each REBUILD3 task.</td>
</tr>
<tr>
<td>SYS010</td>
<td>Contains new db-key information for updating record prefixes. Output: RELOAD4, RELOAD5, and REBUILD3. Input: RELOAD6 One is created for each (RELOAD4 and RELOAD5) task times the number of RELOAD6 tasks, and one is created for each index group (that contains indexes where the member contains UP or OWNER pointers) times the number of RELOAD6 tasks.</td>
</tr>
<tr>
<td>SYS011</td>
<td>Work file containing sorted prefix pointer information for all records. Not created when SORTEXIT is specified. One is created for each RELOAD6 task.</td>
</tr>
</tbody>
</table>
| SYSOF     | Contains new db-key information for updating record prefixes
|           | Output: RELOAD6
|           | Input: RELOAD7
|           | One is created for each RELOAD6 task. |
| DBKEYS    | A file created when the DBKEYS option is specified. Output: RELOAD1 and RELOAD2 One is created for each RELOAD1 and RELOAD2 task. All should be concatenated together when processed. |

### Considerations for Running REORG on VSE

#### Work File Creation and Deletion

REORG on VSE does not support creating and deleting work files. It will create and delete user labels, but the work files will not be created, a VTOC entry will not be created, until the file is opened for output. When REORG processing is complete, work files must be sectionly deleted, or overwritten to reclaim the space. The options: DELETE OLD WORKFILES, CREATE WORKFILES, and DELETEALL, only apply to user labels for the current job. Automatic deletion of work files during the CLEANUP phase will only delete user labels.

**CA DYNAM/D is required to create labels**

The extent of a generated label will have a relative starting track of 1 for the number of tracks based on the primary space value. These labels require CA DYNAM/D to convert them to an actual track address at open time.

If CA DYNAM/D is not installed, labels for work files must be coded sectionly. The recommended procedure in this case, is to use the ESTIMATE FILE SIZES option, which does not use any work files. Code the JCL statements based on the reported file sizes; and include them in all REORG jobs.
SYSIDMS

REORG requires the RORGCTL and RORGJCL files to be defined in SYSIDMS using FILENAME= parameters:

FILENAME=RORGCTL RECFM=F BLKSIZE=8192
FILENAME=RORGJCL RECFM=F BLKSIZE=NNNN LRECL=80
DLBLMOD=ON

Where NNNN is the block size of the JCL file and must be a multiple of 80.

The SYSIDMS DLBLMOD=ON option must be specified to allow for sequential and random processing of the RORGCTL file. Either a DA or SD label m may be used for RORGCTL.

DSMODEL

The only options which apply to z/VSE are: DSN; BLKSIZE; the allocation unit value and primary value for the SPACE option; and the first volume of the VOLSER Option. The rest can be coded, but is ignored.

REORG generates labels using track sizes. If CYL is coded for a space allocation unit, the primary value is converted to tracks, but there is no cylinder alignment. If a block size is coded for an allocation unit, the coded value is used to calculate the number of tracks required. The coded value must match the BLKSIZE value, otherwise the calculated number of tracks may not be accurate.

An example of a DSMODEL follows; note that the primary space value was not coded. This allows REORG to generate the value for each file, using file size estimates. If a primary space value had been coded, this value would be used for all files regardless of the estimated size.

CREATE DSMODEL W*
  DSN 'USERID.EMPDB.WORKFILE.&DD'.
  BLKSIZE 4096
  SPACE TRK
  VOLSER IDMS05 ;

RORGJCL

JCL submitted by REORG is read from the RORGJCL file. This must be a sequential file built on disk and contain all the JECL and JCL statements for the submitted job. However the JECL and some JCL statements cannot be directly copied to this file using normal JCL because POWER will try to interpret these statements when the copy job is run. These statements must be "hidden" from power by changing the first characters as follows:

- * $$ statements must be coded as $ $$ statements
- // JOB statements must be coded as #/ JOB statements
- /* coded as #*
- /& coded as #&
- Any statement starting with a "/" may be coded as starting with a "/#

This is the same method used by the IESINSRT program to hide JCL, which is documented in the z/VSE Administration Section.
The JCL will get stored on disk in the format it is coded. REORG will convert the hiding characters back to their correct values prior to submitting the job to POWER. For example the JCL to copy a job to the RORGJCL file can look like this:

```
// DLBL RORGJCL,'USERID.EMPDB.RORGJCL',1,SD
// EXTENT SYS020,CULLD9,,1,5
// ASSGN SYS020,DISK,VOL=CULLD9,SHR
// EXEC IDCAMS,SIZE=386K
   REPRO INFIL((SYSIPT) -
      OUTFIL(RORGJCL ENV( BLKSZ(4080) RECFM(FB) RECSZ(80)) )
   )
$ $$ JOB JNM=RORGJOB,CLASS=B,DISP=D
$ $$ LST CLASS=R,DEST=(,USERID),JSEP=0
$ $$ PUN CLASS=R,DEST=(,USERID)
#
/ JOB RORGJOB
* JCL THAT REORG SUBMITS
#/ DLBL SYSIDMS,'#SYSIPT'
#/ EXEC IDMSBCF,SIZE=256K
   ECHO=ON JOURNAL=OFF DLBLMOD=ON
   DMCL=IDMSDMCL DBNAME=EMPDEMO
   FILENAME=RORGCTL RECFM=F BLKSIZE=8192
#
REORG;
#*
$ $$ EOJ
/* EOF for REPRO
```

The statements starting with $ $$ JOB and ending with $ $$ EOJ are copied to RORGJCL and are submitted by REORG.

### JCL Considerations

The REORG command must be submitted through the batch command facility. The JCL to execute the facility must include statements to define:

- The standard BCF JCL statements.
- A RORGCTL file. It is a fixed-length, LRECL 8192, unblocked BDAM file that is initialized during setup and read and updated by subsequent and parallel jobs.
- An optional RORGJCL file that references a card-image file containing JCL that is automatically submitted by REORG. This file must be fixed or fixed blocked with an LRECL of 80.
- The files containing the source areas and their dependent areas for the unload phase if dynamic database allocation is not being used to reference them through the source DMCL.
- The files containing the target areas for the RELOAD and REBUILD phases if dynamic database allocation is not being used to reference them through the target DMCL.
- The journal file(s), which can be dummied.
- The work files if dynamic work file allocation is not being used to reference them.

If performing parallel processing, do not specify any syntax that would cause the jobs to serialize.

- Do not specify DISP=OLD or DISP=NEW for database or work files. Specify DISP=SHR instead. If a file is new and must be created, create it prior to submitting any REORG jobs.
DISP=OLD or DISP=NEW is acceptable for a file that is unique to a job. For example SYSLST, SYSOUT, or SYSJRNL are unique to a job. Sort files such as SORTWK01 and SORTMSG are also unique to a given job.

If you define the same DDNAMEs for the source and target database files, the database cannot be unloaded and reloaded in the same job. You must halt processing after the unload phase and then restart processing ensuring that the target files and not the source files are being referenced. Follow these general steps:

- Submit JCL for the unload phase. The source database files must be referenced in the REORG JCL and in JCL submitted from RORGJCL, unless using dynamic database file allocation.
- Use the STOP AFTER UNLOAD option to halt the REORG process after the unload phase.
- After all unload jobs have ended and the unload phase has completed, format the target database files if they have not already been formatted.
- Submit JCL for the reload phases. The submitted job and JCL submitted from RORGCTL must reference the target database files, unless using dynamic database file allocation.
- The submitted job must say STOP AFTER CLEANUP or STOP AFTER RELOAD to allow tasks in the reload phases to run.

Work files can be manually or dynamically allocated. Considering the number of work files needed, the use of dynamic work file allocation is recommended. A CREATE DSMODEL must be coded in the SETUP job, or in the job that allocates the files to use dynamic allocation.

⚠️ **Note:** When using CREATE DSMODEL statements, it is recommended that you force the batch command facility to terminate if an error is encountered. You do this by specifying a SET OPTIONS statement with the ON ERROR END clause. By forcing a termination, it ensures that REORG will not execute with incomplete data set models due to an error in a CREATE DSMODEL statement.

When manually allocating work files, you should run a preliminary REORG with the STOP AFTER SETUP option. This produces a report of all required work files. You must allocate these files prior to restarting the REORG process and include the necessary DD statements with DISP=SHR in the JCL that is submitted to restart processing and in the JCL in the RORGJCL file.

The RORGCTL file should be created prior to submitting the first REORG job, unless STOP AFTER SETUP is specified. REORG jobs running in parallel must specify DISP=SHR on this file, or they will serialize with each other on the DSNAME. If processing is stopped after SETUP, the SETUP job can create the file, but subsequent jobs must specify DISP=SHR.

⚠️ **Note:** For more information about the generic JCL used to execute the batch command facility, see the section for your operating system in this section.
Examples

Example 1

The following example reorganizes the EMPDEMO database. To speed execution, the data is divided into three slices that can be processed in parallel. All work files are dynamically allocated because data set model information is present. Because no STOP AFTER parameter is specified, the reorganization runs to completion.

Create dsmodel w* ...
Reorg segment empdemo using empss01
reload into empdemo dmcl newdmcl
divide processing 3 ways
reuse workfile;

Example 2

The following example separates the setup phase from the later phases. After executing setup multiple times, database reorganization is begun using the second REORG statement. Again, work files are allocated dynamically. The SUBMIT directive submits multiple jobs to to process the work in parallel.

Create dsmodel w* ...
Reorg segment empdemo using empss01
reload into empdemo dmcl newdmcl
divide processing 3 ways
reuse workfiles
stop after setup;
Reorg submit stop after cleanup;

Example 3

The following example restarts a reorganization that was interrupted by a system failure. It too submits multiple jobs to process the work.

Reorg submit;

Example 4

The following example shows an extra DSMODEL being specified for a DBKEYS file. The DBKEYS file DDNAME starts with WD so it matches on the WD* DSMODEL. The SPACE and BLKSIZE attributes are taken from this model and remaining attributes are taken from the W* DSMODEL because of the FROM syntax. The remaining REORG work files match on the W* DSMODEL and take attributes directly from it.

CREATE DSMODEL W*
DSN 'IDMS.REORG.WORKFILE.&DD'.
UNIT SYSDA
SPACE TRK(5,5)
BLKSIZE 12004
;
CREATE DSMODEL WD*
FROM W*
SPACE TRK(2,2)
BLKSIZE 16000
;
REORG SEGMENT REOGSMAL USING REOGSUB
RELOAD INTO REOGSMAL
DIVIDE PROCESSING 4 WAYS
STOP AFTER UNLOAD
CREATE ALL WORKFILES
GENERATE DBKEYS
NOTIFY 10
;

Example 5

The following example shows the syntax required for a job submitted through RORGJCL. DSMODEL and other options are not coded because they are taken from the control file and are not needed by this job. A job with this syntax can also be submitted sectionly, and it also uses existing options from the control file and looks for tasks to process.

REORG;

Sample Output

When a REORG job completes, it prints a status report of the current REORG process. This report can also be produced using the REORG STATUS REPORT ONLY option. The report is made up of several sections, which contains an analysis of the current REORG operation and a status of the REORG tasks.

REORG Status Report - Section 1

The first section of the REORG Status Report identifies an overall status of the REORG operation.

******************************************************************************
* Identifying time stamp: 2005-11-17-13.44.06.380141 *
* Unload subschema=EMPTSS01 Unload segment=EMPDEMO Unload DMCL=EMPTDMCL *
* Reload subschema=EMPTSS01 Reload segment=EMPBDEMO Reload DMCL=EMPBDMCL *
******************************************************************************

Options in effect
-----------------
Divide processing 3 ways Notify interval=1 Job submission=YES SHARE=YES
STOP AFTER CLEANUP CREATE ALL WORKFILES reuse workfiles=NO SORTEXIT=NO
Overflow Percentage=10 Concurrent jobs=5 GENERATE DBKEYS FILES=YES

Current status
---------------
SETUP=completed UNLOAD=completed RELOAD=completed CLEANUP=completed

REORG Status Report - Section 2

The next section identifies all the records and sets being processed by the REORG operation. The sequence numbers are assigned by REORG and are used to identify RECORDS and SETS in REORG work files. They have no relation to record or index IDs that are assigned to database records.

System indexes are assigned to groups. All indexes in the same group are offloaded and rebuilt together. Indexes are assigned to the same group when they share a common page range.

Records and Sets
-----------------
Set=EMP-NAME-NDX     Sequence=1     Index group=1
Set=JOB-TITLE-NDX    Sequence=2     Index group=2
Set=SKILL-NAME-NDX   Sequence=3     Index group=2
Set=COVERAGE-CLAIMS   Sequence=4
Set=DEPT-EMPLOYEE    Sequence=5
Set=EMP-COVERAGE     Sequence=6
Set=EMP-EXPERTISE    Sequence=7
The next section of the report identifies unique page ranges within the source and target databases. All records in the same page range share one or more pages and do not overlap with records outside the page range. These ranges are used by REORG setup to determine how to slice up a database.

Page ranges that contain only indexes are also identified.

Areas and Slices

---

Unload area=EMPDEMO.EMP-DEMO-REGION  low page=175001 high page=175400
Records only area subrange  low page=175021 high page=175400
Index only area subrange  low page=175002 high page=175017
Unload area=EMPDEMO.INS-DEMO-REGION  low page=175401 high page=175600
Records only area subrange  low page=175421 high page=175600
Records only area subrange  low page=175402 high page=175419
Unload area=EMPDEMO.ORG-DEMO-REGION  low page=175601 high page=175800
Records only area subrange  low page=175621 high page=175800
Index only area subrange  low page=175602 high page=175619
Reload area=EMPBDEMO.EMP-DEMO-REGION  low page=275001 high page=275800
Records only area subrange  low page=275041 high page=275800
Index only area subrange  low page=275002 high page=275033
Reload area=EMPBDEMO.INS-DEMO-REGION  low page=276001 high page=276400
Records only area subrange  low page=276041 high page=276400
Records only area subrange  low page=276002 high page=276037
Reload area=EMPBDEMO.ORG-DEMO-REGION  low page=277001 high page=277400
Records only area subrange  low page=277041 high page=277400
Index only area subrange  low page=277002 high page=277037

REORG Status Report - Section 4

The next section reports the portions of each page range assigned to each slice.

Unload slice=1
Unload slice range  low page=175021 high page=175147
Unload slice range  low page=175421 high page=175520
Unload slice range  low page=175402 high page=175419
Unload slice range  low page=175621 high page=175720
Unload slice=2
Unload slice range  low page=175148 high page=175274
Unload slice range  low page=175521 high page=175600
Unload slice range  low page=175721 high page=175800
Unload slice=3
Unload slice range  low page=175275 high page=175400
Reload slice=1
Reload slice range  low page=275041    high page=275294
Reload slice range  low page=276281    high page=276400
Reload slice range  low page=277161    high page=277280
Reload slice=2
Reload slice range  low page=275295    high page=275547
Reload slice range  low page=276041    high page=276160
Reload slice range  low page=276002    high page=276037
Reload slice range  low page=277281    high page=277400
Reload slice=3
Reload slice range  low page=275548    high page=275800
Reload slice range  low page=276161    high page=276280
Reload slice range  low page=277041    high page=277160

**REORG Status Report - Section 5**

The next section describes the status of each task and the work files assigned to the task. The data set name is displayed only if dynamic work file allocation is used.

### Tasks and Work File Usage

<table>
<thead>
<tr>
<th>Task#</th>
<th>Task</th>
<th>type</th>
<th>slice</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>UNLOAD</td>
<td>S</td>
<td>1</td>
<td>completed</td>
</tr>
<tr>
<td></td>
<td>SYS002 O/P</td>
<td>DNAME=WU00001</td>
<td>DSN=USERA01.EMPDEMO.WORKFILE.WU00001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYS002 O/P</td>
<td>DNAME=WU00002</td>
<td>DSN=USERA01.EMPDEMO.WORKFILE.WU00002</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYS003 O/P</td>
<td>DNAME=WU00003</td>
<td>DSN=USERA01.EMPDEMO.WORKFILE.WU00003</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>UNLOAD</td>
<td>S</td>
<td>2</td>
<td>completed</td>
</tr>
<tr>
<td></td>
<td>SYS002 O/P</td>
<td>DNAME=WU00004</td>
<td>DSN=USERA01.EMPDEMO.WORKFILE.WU00004</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYS002 O/P</td>
<td>DNAME=WU00005</td>
<td>DSN=USERA01.EMPDEMO.WORKFILE.WU00005</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYS003 O/P</td>
<td>DNAME=WU00006</td>
<td>DSN=USERA01.EMPDEMO.WORKFILE.WU00006</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>UNLOAD</td>
<td>S</td>
<td>3</td>
<td>completed</td>
</tr>
<tr>
<td></td>
<td>SYS002 O/P</td>
<td>DNAME=WU00007</td>
<td>DSN=USERA01.EMPDEMO.WORKFILE.WU00007</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYS002 O/P</td>
<td>DNAME=WU00008</td>
<td>DSN=USERA01.EMPDEMO.WORKFILE.WU00008</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYS002 O/P</td>
<td>DNAME=WU00009</td>
<td>DSN=USERA01.EMPDEMO.WORKFILE.WU00009</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYS003 O/P</td>
<td>DNAME=WU00012</td>
<td>DSN=USERA01.EMPDEMO.WORKFILE.WU00012</td>
<td></td>
</tr>
</tbody>
</table>

**REORG Status Report - Section 6**

The next section shows the work file summary. This example shows the general-purpose files used during the unload and reload phases. Other sections show index files, sorted data files, and DBKEYS files.

All work file summary reports show the estimated size of each file and the attributes used in determining those sizes. BPT is Blocks Per Track; TPC is Tracks Per Cylinder; and *3390* indicates that attributes were based on a generic 3390 device, as opposed to a specific volume.

An asterisk (*) following a DDNAME indicates that the file was created by a REORG job executing as part of the current operation and therefore will be deleted automatically during cleanup.

### Unload/Reload work file summary

<table>
<thead>
<tr>
<th>DDname</th>
<th>DSN</th>
<th>Estimated Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>WU00001</td>
<td>*USERA01.EMPDEMO.WORKFILE.WU00001</td>
<td>1 Trk</td>
</tr>
<tr>
<td></td>
<td>Estimate based on: VOLSER=<em>3390</em>  BLKSIZE=27998  BPT=02  TPC=15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UNIT=SYSDA  SPACE=Trk  PRI=1  SEC=1</td>
<td></td>
</tr>
<tr>
<td>WU00002</td>
<td>*USERA01.EMPDEMO.WORKFILE.WU00002</td>
<td>1 Trk</td>
</tr>
<tr>
<td></td>
<td>Estimate based on: VOLSER=<em>3390</em>  BLKSIZE=27998  BPT=02  TPC=15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UNIT=SYSDA  SPACE=Trk  PRI=1  SEC=1</td>
<td></td>
</tr>
</tbody>
</table>
REORG Status Report - Section 7

The next section is a recap of tasks that have completed in task order. It shows when the task was started, when it was completed, and the job that processed the task.

******************************************************************************
*                                                                            *
*                  Recap of TASK activity                                    *
*                                                                            *
******************************************************************************

Task#=1 Task=UNLOAD type=S slice=1 status=completed
jobname=USERA01Y jobid=JOB01048
UT019000 Task 1 UNLOAD slice 1 starting at 2005-11-17-13.44.35.255392
UT019003 Task 1 UNLOAD slice 1 completed at 2005-11-17-13.44.35.884474

Task#=2 Task=UNLOAD type=S slice=2 status=completed
jobname=USERAXXA jobid=JOB05591
UT019000 Task 2 UNLOAD slice 2 starting at 2005-11-17-13.44.35.752587
UT019003 Task 2 UNLOAD slice 2 completed at 2005-11-17-13.44.36.077111

REORG Status Report - Section 8

The next section shows the tasks processed by each job in job name sequence.

******************************************************************************
*                                                                            *
*                  Recap of JOB activity                                     *
*                                                                            *
******************************************************************************

jobname=USERA01Y jobid=JOB01048
UT019000 Task 1 UNLOAD slice 1 starting at 2005-11-17-13.44.35.255392
UT019003 Task 1 UNLOAD slice 1 completed at 2005-11-17-13.44.35.884474

jobname=USERAXXA jobid=JOB05591
UT019000 Task 2 UNLOAD slice 2 starting at 2005-11-17-13.44.35.752587
UT019003 Task 2 UNLOAD slice 2 completed at 2005-11-17-13.44.36.077111

jobname=USERAXXB jobid=JOB05592

UTE019000 Task 3 UNLOAD slice 3 starting at 2005-11-17-13.44.35.826364
UTE019003 Task 3 UNLOAD slice 3 completed at 2005-11-17-13.44.36.103013

REORG Status Report - Section 9

The last section shows task activity in chronological order. When completed, it shows the elapsed time from when REORG was started to when it completed.

******************************************************************************
*                                                                            *
*                  Recap of activity by time of day                         *
*                                                                            *
******************************************************************************

USERA01Y JOB01048 UTE019000 Task 1 UNLOAD slice 1 starting at 2005-11-17-13.44.35.255392
USERAXXA JOB05591 UTE019000 Task 2 UNLOAD slice 2 starting at 2005-11-17-
13.44.35.752587
USERAXXB J0805592 UT019000 Task 3 UNLOAD slice 3 starting at 2005-11-17-
13.44.35.826364
USERAXXC J0805593 UT019002 Task 4 UNLOAD index group 1 starting at 2005-11-17-
13.44.35.866943
USERA01Y J0801048 UT019003 Task 1 UNLOAD slice 1 completed at 2005-11-17-
13.44.35.884474
USERAXXC J0805593 UT019005 Task 4 UNLOAD index group 1 completed at 2005-11-17-
13.44.35.985660
USERAXX A J0805591 UT019003 Task 2 UNLOAD slice 2 completed at 2005-11-17-
13.44.36.077111
USERAXXB J0805592 UT019003 Task 3 UNLOAD slice 3 completed at 2005-11-17-
13.44.36.103013
USERA01Z J0801069 UT019000 Task 5 RELOAD1 slice 1 starting at 2005-11-17-
13.46.56.274321
USERA01Z J0801069 UT019000 Task 5 RELOAD1 slice 1 completed at 2005-11-17-
13.46.57.381739
USERAXXA J0805622 UT019005 Task 22 REBUILD4 index group 1 completed at 2005-11-17-
13.47.21.060599
USERA01Z J0801069 UT019005 Task 23 REBUILD4 index group 2 completed at 2005-11-17-
13.47.21.228884
USERA01Z J0801069 UT019014 All REORG phases have been completed at 2005-11-17-
13.47.48.448372
USERA01Z J0801069 UT019015 REORG elapsed time: 0 days 0 hours 3 minutes 42 seconds

**RELOAD Statistics Report - Section 1**

The following example shows standard database statistics as returned from an ACCEPT DATABASE
STATISTICS command. The values reflect the processing done by the current RELOAD1 or RELOAD2
task.

* UT005002 DATABASE LOAD STATISTICS
* DATABASE LOADED ON 01/15/08 AT 21:24:53
* PAGES READ .................. 62
* PAGES WRITTEN ............... 59
* PAGES REQUESTED ............. 86
* CALC RCDS IN TARGET PAGE ..... 268
* CALC RCDS OVERFLOWED ....... 0
* VIA RCDS IN TARGET PAGE ...... 3,869
* VIA RCDS OVERFLOWED ........ 0
* LINES REQUESTED BY IDMS ...... 1,742
* RCDS MADE CURRENT OF R/U ..... 4,137
* CALLS TO IDMS ............... 4,144
* FRAGMENTS STORED ........... 0
* RECORDS RELOCATED .......... 0

**RELOAD Statistics Report - Section 2**

The following example shows additional statistics produced by REORG when reloading a database.
The values reflect the processing done by the current RELOAD1 or RELOAD2 task.

**RELOAD STATS**
* RCDS READ .................. 4,380
* RCDS STORED IN DATABASE ... 4,137
* RCDS STORED ON TARGET PAGE .. 3,998
* RCDS STORED TARGET PERCENT ... 96
* RCDS WRITTEN TO OVERFLOW .... 243
* RCDS CACHED .................. 357
* RCDS STORED FROM CACHE ....... 139
* RCDS OVERFLOW FROM CACHE ...... 218
* CACHE STORAGE USED ........... 16,384
* CACHE STORAGE LIMIT .......... 16,384

The fields reported are the following:
• **RCDS READ**
  The total database records read from a work file. This does not include pointer and other work records.

• **RCDS STORED IN DATABASE**
  The number of database records stored in the database.

• **RCDS STORED ON TARGET PAGE**
  The number of database records stored on the intended target page.

• **RCDS STORED TARGET PERCENT**
  The percentage of database records stored on their target page.

• **RCDS WRITTEN TO OVERFLOW**
  The number of database records written to a SYSOF2 file, to be processed by the RELOAD2 overflow task.

• **RCDS CACHED**
  The number of database records written to the memory cache.

• **RCDS STORED FROM CACHE**
  The number of database records that were written to memory cache and then stored in the database.

• **RCDS OVERFLOW FROM CACHE**
  The number of database records that were written to memory cache and then written to the overflow work file.

• **CACHE STORAGE USED**
  The amount of allocated cache storage at the end of task. If this is the same as the storage limit, the cache reached its limit and records may have been written to the overflow file.

• **CACHE STORAGE LIMIT**
  The maximum storage allowed for the overflow cache. This can be changed using the OVERFLOW CACHE option.

---

**DBKEYS File Layout**

This section describes the contents of the record written to a DBKEYS cross reference file. The layout matches the DBKEYS file generated by CA IDMS/DB Reorg.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Element Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLDDBKEY</td>
<td>PIC S9(8) COMP</td>
<td>Old db-key</td>
</tr>
<tr>
<td>NEWDBKEY</td>
<td>PIC S9(8) COMP</td>
<td>New db-key</td>
</tr>
<tr>
<td>SR3DBKEY</td>
<td>PIC S9(8) COMP</td>
<td>SR3 db-key if the old record was relocated</td>
</tr>
<tr>
<td>RECORDID</td>
<td>PIC S9(4) COMP</td>
<td>Record ID in the source subschema</td>
</tr>
<tr>
<td>SR3RECID</td>
<td>PIC S9(4) COMP</td>
<td>Original record ID if relocated</td>
</tr>
</tbody>
</table>
The logical record length is 16 bytes. The BLKSIZE is determined by what is coded in the JCL or matching CREATE DS MODEL statement.

Each task that loads data generates a DBKEYS file for the data it loads. Depending on their intended use, these files may need to be merged together and sequenced before being used by another application.

RESTORE

The RESTORE utility restores one or more areas in a database by copying back the contents of a file created by the BACKUP statement.

This article describes the following information:

- Authorization (see page 237)
- Syntax (see page 237)
- Parameters (see page 237)
- Usage (see page 238)
- JCL Considerations (see page 239)
- Examples (see page 239)
- Sample Output (see page 239)

Authorization

To restore an area, a file, or a segment, you need DBAWRITE access privileges on the area.

Syntax

```
RESTORE
  AREA segment-name.area-name
  FILE segment-name.file-name
  SEGMENT segment-name

BY AREA
BY FILE
```

Parameters

- **AREA**
  Directs the RESTORE utility to restore one or more areas.

- **segment-name**
  Specifies the name of the segment associated with the area.
• **area-name**
  Specifies the name of the area.

• **FILE**
  Directs the RESTORE utility to restore one or more files.

• **segment-name**
  Specifies the name of the segment associated with the file.

• **file-name**
  Specifies the name of the file.

• **SEGMENT segment-name**
  Specifies the name of the segment to be restored.

• **BY AREA**
  Specifies that each area defined within the segment is to be restored. AREA is the default.

• **BY FILE**
  Specifies that each file within the segment is to be restored.

### Usage

**How to submit the RESTORE statement**

You submit the RESTORE statement only through the batch command facility. When submitting RESTORE statements, the batch command facility must be run in local mode.

**Vary areas offline**

Before running the RESTORE statement, vary all areas being restored offline to all DC/UCF systems. This will prevent all other jobs from accessing the areas until RESTORE processing is completed.

**Restoring by area requires locks on areas**

CA IDMS/DB locks an area before restoring it. If the area is already locked, it is not restored and the RESTORE operation will terminate with an error. No more areas are restored. It is possible that a local mode application abended without releasing an area lock. In this case, use the UNLOCK statement to unlock the area.

**Restoring by file does not lock areas**

When you restore by file, CA IDMS/DB does not lock the associated area(s).

**RESTORE the same object you backed up**

If you backed up by area, restore by area. If you backed up by file, restore by file.

**Restoring from IDMSDUMP**
RESTORE can only restore files produced by the BACKUP utility. RESTORE cannot restore files produced by the 10.2 IDMSDUMP utility program. You must use the 10.2 IDMSRSTR utility program to restore files produced by IDMSDUMP.

JCL Considerations

When you submit a RESTORE statement through the batch command facility, the JCL to execute the facility must include statements to define the file(s) containing the areas to be restored.

⚠️ **Note:** For more information about the generic JCL used to execute the batch command facility for your operating system, see z/OS JCL (see page 355) and z/VSE JCL (see page 391).

Examples

Restoring by area

The following example directs the RESTORE utility to restore three database areas.

```sql
restore area empdemo.emp-demo-region,
        empdemo.org-demo-region,
        empdemo.ins-demo-region;
```

Restoring by file

The following example directs the RESTORE utility to restore three database files.

```sql
restore file empdemo.empdemo.
        empdemo.orgdemo,
        empdemo.insdemo;
```

Restoring by segment

The following example directs the RESTORE utility to restore all areas in the empdemo segment.

```sql
restore segment empdemo;
```

Sample Output

**Restoring a database by area**

The following listing is generated after the statements in the "Restoring a Database by Area" example are successfully executed.
RESTORE AREA EMPDEMO.EMP-DEMO-REGION,
   EMPDEMO.ORG-DEMO-REGION,
   EMPDEMO.INS-DEMO-REGION;

UT015006 BACKUP file created on yyyy-mm-dd-hh.mm.ssffffff
UT000038 Starting RESTORE of area EMPDEMO.EMP-DEMO-REGION
UT000040 RESTORE complete
UT000038 Starting RESTORE of area EMPDEMO.ORG-DEMO-REGION
UT000040 RESTORE complete
UT000038 Starting RESTORE of area EMPDEMO.INS-DEMO-REGION
UT000040 RESTORE complete
Status = 0        SQLSTATE = 00000

Restoring a database by file

The following listing is generated after the statements in the "Restoring a Database by File" example shown previously are successfully executed.

IDMSBCF nn.
n CA IDMS Batch Command Facility
   mm/dd/yy PAGE 4

RESTORE FILE EMPDEMO.EMPDEMO,
   EMPDEMO.ORGDEMO,
   EMPDEMO.INSDEMO;

UT015006 BACKUP file created on yyyy-mm-dd-hh.mm.ssffffff
UT000039 Starting RESTORE of file EMPDEMO.EMPDEMO
UT000040 RESTORE complete
UT000039 Starting RESTORE of file EMPDEMO.ORGDEMO
UT000040 RESTORE complete
UT000039 Starting RESTORE of file EMPDEMO.INSDEMO
UT000040 RESTORE complete
Status = 0        SQLSTATE = 00000

Restoring a database by segment

The following listing is generated after the statements in the "Restoring a Database by Segment" example are successfully executed.

IDMSBCF nn.
n CA IDMS Batch Command Facility
   mm/dd/yy PAGE 6

RESTORE SEGMENT EMPDEMO;

UT015006 BACKUP file created on yyyy-mm-dd-hh.mm.ssffffff
UT000038 Starting RESTORE of area EMPDEMO.EMP-DEMO-REGION
UT000040 RESTORE complete
UT000038 Starting RESTORE of area EMPDEMO.INS-DEMO-REGION
UT000040 RESTORE complete
UT000038 Starting RESTORE of area EMPDEMO.ORG-DEMO-REGION
UT000040 RESTORE complete
Status = 0        SQLSTATE = 00000
RESTRICTURE

The RESTRUCTURE utility modifies record occurrences to match new schema specifications.

Using the RESTRUCTURE utility, you can:

- Insert new data items anywhere in a record
- Delete existing data items
- Change the length and position of data items
- Change the format of a record from fixed length to variable length or from variable length to fixed length
- Compress or uncompress a record
- Add pointers for new or existing sets
- Delete pointers from existing sets
- Add or delete prior or owner pointers for existing sets

This article describes the following information:

- Authorization (see page 241)
- Syntax (see page 241)
- Parameters (see page 242)
- Usage (see page 243)
- JCL Considerations (see page 245)
- Examples and Sample Output (see page 245)
- Callable Restructure Utility (see page 247)

Authorization

<table>
<thead>
<tr>
<th>To</th>
<th>You need this privilege</th>
<th>On</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restructure an area</td>
<td>DBAWRITE</td>
<td>The area</td>
</tr>
<tr>
<td>Restructure a segment</td>
<td>DBAWRITE</td>
<td>All areas of the segment</td>
</tr>
</tbody>
</table>

Syntax

```
|►►| RESTRUCTURE | SEGMENT segment-name |
|   |             |                  |
|   | USING subschema-name |
|   |                  |
```
Parameters

- **db-name**
  Identifies the name of the database to be restructured.

- **USING**
  Specifies the subschema that defines all the records and sets to be restructured.

- **subschema-name**
  The name of a subschema compiled under a schema that describes the database before restructuring.

- **CONTINUE**
  Specifies whether processing is to continue if an error condition is detected during processing. By default, if you do not specify YES, processing stops when an error is detected.

- **YES**
  Directs processing to continue when an error is detected.

- **NO**
  Directs processing to stop when an error is detected. NO is the default.

- **RSTTMOD**
  Specifies the base restructuring table that defines the changes to be made in the database. By default, if you do not specify a base restructuring table, IDMSRSTT is used.

- **rstt-modname**
  The name of the base restructuring table. The default base restructuring table is IDMSRSTT.

- **AREA**
  Restricts processing to one or more specified areas. The parameter **segment-name** qualifies which area the changes need to be made in. By default, if you do not specify one or more areas, all areas in the specified segment that contain occurrences of the records or members of the sets being restructured are processed.

- **area-name**
  The name of an area.
Usage

How to submit the RESTRUCTURE statement

You submit the RESTRUCTURE statement only through the batch command facility. When submitting RESTRUCTURE statements, you must run the batch command facility in local mode.

The base restructuring table

The RESTRUCTURE statement modifies record occurrences according to the specifications in a base restructuring table. You assemble the base restructuring table from IDMSRSTT macro statements that define the changes to be made. Typically, you use the IDMSRSTC utility to generate the IDMSRSTT macro statements; however, the statements can also be coded sectionly.

Note: For more information, see IDMSRSTT Macro Statements (see page 478).

The spill file

If you specify a set-name in the SETPTR statement of IDMSRSTT during RESTRUCTURE processing, a spill file of work records that describes new pointers being added to database records is generated. If the database restructure includes the addition of new prior pointers to existing sets, you use the spill file as input to the RESTRUCTURE CONNECT utility statement. The information in the spill file and the specifications in the base restructuring table are used during RESTRUCTURE processing to connect the new prior pointers.

Database keys of restructured records

During RESTRUCTURE processing all changes to the database are made in place (that is, no unload /reload occurs). As a result, database keys are not changed by a restructure operation.

Back up the database

Back up the database before performing a restructure operation. If the database restructure is unsuccessful, you can then restore the database from the backup copy.

Remove logically deleted records

You cannot use the RESTRUCTURE utility statement to make changes that will modify the prefix portion of a record if there are logically deleted records in the database area to be modified. To ensure there are no logically deleted records, use the CLEANUP utility to erase them before performing a restructure operation.

Restructure ready mode

All areas to be processed are readied in exclusive update mode for RESTRUCTURE processing.

Modifying CALC and sort keys

During a restructure operation, you should not modify CALC and sort keys.
Native VSAM data sets

You cannot use RESTRUCTURE to restructure records in native VSAM data sets.

New pointers

New pointers in an owner record are initialized to the database key of the owner record (indicating an empty set). In a member record, new pointers are initialized to -1.

To connect new pointers for existing sets, use the RESTRUCTURE CONNECT utility statement. To connect pointers for new sets, you must run a user-written program after the restructure process is complete.

Consideration when using data compression routines

If the named subschema references a data compression routine and a compressed record is involved in the restructure (for example, changing the record to fixed length), the subschema will be modified. To modify the subschema, it must be nonreentrant. If necessary, relink the subschema as nonreentrant and run it from another load library.

If the subschema resides in a dictionary load area, this is not a consideration.

Restructuring a database

To restructure a database, follow these steps:

1. **Back up the database.**
2. **Compile a new schema** that describes the database after restructuring. The new schema must have a different name or version number from the schema that describes the existing database.
3. **Execute the IDMSRSTC utility** to generate the IDMSRSTT macro statements that define the changes to be made to the database. Verify that the statements are correct; make any necessary modifications.
4. **Assemble the base restructuring table.**
5. **Link edit the base restructuring table.**
6. **Execute the RESTRUCTURE utility statement** to restructure the database. Use a subschema that was compiled under the old schema.
7. **Compile a subschema under the new schema.** Give the new subschema a temporary name to distinguish it from the old subschema.
   If no new pointers have been added to existing sets, skip to step 10.
8. **Execute the RESTRUCTURE CONNECT utility statement** to connect the new prior or owner pointers for existing sets in the restructured database. Use a subschema that was compiled under the new schema.

9. **Validate the restructure** using IDMSDBAN, CA OLQ, CA Culprit, or some other retrieval program.

10. **Recompile, under the new schema, all subschemas that use the changed records or sets.**

11. **Alter all access modules referencing changed records or owner and member records of changed sets.**

12. **Drop and recreate SQL views of records whose element descriptions have changed.**

**JCL Considerations**

When you submit a RESTRUCTURE statement through the batch command facility, the JCL to execute the facility must include statements to define:

- The database file(s) that map to the area(s) to be processed.
- The file containing the assembled base restructuring table.
- The spill file to be used as input to the RESTRUCTURE CONNECT utility statement. The size of the spill file should be a multiple of 40 with a maximum size of 32,760.

⚠️ **Note:** For more information about the generic JCL used to execute the batch command facility, see the Batch Command Facility (z/OS) (see page 355) or z/VSE JCL (see page 389) topics.

**Examples and Sample Output**

**Adding set pointers**

The following example directs the RESTRUCTURE utility to use the base restructuring table, LRDKRSTT, to add prior pointers to the owners and members of set A-B and owner pointers to the members of set A-B.

*Input to RESTRUCTURE*

```
RESTRUCTURE dempdemo using restr01
  rsttmod lrdkrstt continue yes;
```

*Output from RESTRUCTURE*

When the RESTRUCTURE operation is completed, the following listing is generated.
RESTRUCTURE EMPDEMO USING RESTR01
RSTTMOD LRDKRSTT CONTINUE YES;
0UT000038 Starting Restructure of area EMPDEMO.EMP-DEMO-REGION
UT000041 Completed processing of area EMPDEMO.EMP-DEMO-REGION, Pages read=10 Records read=6
UT011023 Record name AA Found=2 Changed=2 Ldel=0
0UT000038 Starting Restructure of area EMPDEMO.INS-DEMO-REGION
UT000041 Completed processing of area EMPDEMO.INS-DEMO-REGION, Pages read=10 Records read=3
UT011023 Record name BB Found=3 Changed=3 Ldel=0
Status = 0

Connecting prior and owner pointers

To complete this restructure operation, the RESTRUCTURE CONNECT utility is executed to connect the prior and owner pointers in all occurrences of set A-B after restructuring records A and B.

Input to RESTRUCTURE CONNECT

restructure connect segment empdemo using EMPSS01
      rsttmod lrdkrstt continue yes;

Output from RESTRUCTURE CONNECT

When the RESTRUCTURE CONNECT operation is completed, the following listing is generated.

RESTRUCTURE CONNECT SEGMENT EMPDEMO USING EMPSS01
RSTTMOD LRDKRSTT CONTINUE YES;
0UT000038 Starting Connect of area EMPDEMO.EMP-DEMO-REGION
UT000041 Completed processing of area EMPDEMO.EMP-DEMO-REGION, Pages read=10 Records read=6
UT011023 Record name AA Found=2 Changed=2 Ldel=0
0UT000038 Starting Connect of area EMPDEMO.INS-DEMO-REGION
UT000041 Completed processing of area EMPDEMO.INS-DEMO-REGION, Pages read=10 Records read=3
UT011023 Record name BB Found=3 Changed=3 Ldel=0
Status = 0

More Information

- For more information about changing a database definition, see Administrating CA IDMS Database
  (https://docops.ca.com/display/IDMS/Administrating+CA+IDMS+Database).

- For more information about CA IDMS/DB physical database characteristics, see Administrating CA IDMS Database
  (https://docops.ca.com/display/IDMS/Administrating+CA+IDMS+Database).

- For more information about coding IDMSRSTT macro statements, see IDMSRSTT Macro Statements (see page 478).
Callable Restructure Utility

The Callable Restructure Utility (IDMSCRSU) is a version of the CA IDMS RESTRUCTURE utility that is callable from a user-written program. It uses the standard base restructuring table to control the restructure of the data portion of a database record.

Note: For more information about assembling a base restructuring table, see IDMSRSTT Macro Statements (see page 478).

IDMSCRSU is called with the following register values and parms:

On Entry:
- R1 points to a four-word parmlist.
- R13 should point to a 36-word save and work area.
- R14 contains the return address.
- R15 contains the entry point address.

Considerations

IDMSCRSU does not support changes to pointers or calling database procedures. The assumption is that the caller has obtained the database record in the old subschema format and now wants to store or modify a record in the new subschema format.

To save a search of the IDMSRSTT base restructuring table on every call, the caller can save the RREC address returned in R0 after the first call. This address can then be passed instead of the IDMSRSTT address on all subsequent calls for the same record type.

A 36-word register save area and work area is expected to be passed in R13. If called from a DC assembler program, you can use a 
"#CHKSTK =36" instruction to ensure there is enough room in the stack.

Parameters

- **parm-1**
  The address of the FSR (SR51) control block from the "old"subschema. This block should describe the record as it exists before the restructure takes place.

- **parm-2**
  The address of the IDMSRSTT base restructuring table that describes the changes being made to the database record. This table is generated from the IDMSRSTT macro. (See Section B.) The table is searched for the RREC block that matches the FSR name passed in parm-1.
  - or -
  The address of the RREC control block in the IDMSRSTT table. (No search is done.)
- **parm-3**
  The address of the data record to be restructured. Only the data should be passed, not pointers or internal fields.

- **parm-4**
  The address of a buffer where the restructured record will be returned. It should be large enough to hold the maximum size of the record being restructured as described in the 'new' subschema.

  On exit:
  - R0 contains the address of the IDMSRSTT RREC block for the record just restructured, if found.
  - R15 contains one of the following return codes:
    - 0: Call successful - The output buffer contains the restructured record.
    - 4: FIELD=ALL was specified for the record in the IDMSRSTT table. There were no changes to the data. The output buffer does not contain a copy of the record; the caller should use the input record image.
    - 8: The record was not found in the IDMSRSTT table.

### RESTRUCTURE CONNECT

The RESTRUCTURE CONNECT utility statement connects new prior and owner pointers in existing sets. You execute the RESTRUCTURE CONNECT statement after executing the RESTRUCTURE statement as part of a database restructure operation. The RESTRUCTURE CONNECT statement uses the specifications in the base restructuring table and the information in the spill file generated by the RESTRUCTURE statement to make the pointer connections.

This article describes the following information:

- Authorization (see page 248)
- Syntax (see page 249)
- Parameters (see page 249)
- Usage (see page 250)
- JCL Considerations (see page 250)
- Examples and Sample Output (see page 251)

### Authorization

<table>
<thead>
<tr>
<th>To</th>
<th>You need this privilege</th>
<th>On</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect pointers in an area</td>
<td>DBAWRITE</td>
<td>The area</td>
</tr>
<tr>
<td>Connect pointers in a segment</td>
<td>DBAWRITE</td>
<td>All areas of the segment</td>
</tr>
</tbody>
</table>
Syntax

RESTRUCTURE CONNECT
  SEGMENT segment-name
  DBNAME db-name

USING subschema-name

CONTINUE
  YES
  NO

RSTTMOD
  rstt-modname
  IDMSRSTT

AREA
  area-name

Parameters

- **db-name**
  Identifies the database name whose segments include the area or areas to be processed.

- **USING**
  Specifies the subschema that defines the records and sets being restructured.

- **subschema-name**
  The name of a subschema compiled under a schema that describes the database after restructuring.

- **CONTINUE**
  Specifies whether processing is to continue if an error condition is detected during processing. By default, if you do not specify YES, RESTRUCTURE CONNECT will not continue processing when an error is detected.

- **YES**
  Specifies that processing should continue when an error is detected.

- **NO**
  Specifies that processing should stop when an error is detected. NO is the default.

- **RSTTMOD**
  Specifies the base restructuring table that defines the changes in the database being made by the RESTRUCTURE and RESTRUCTURE CONNECT statements. By default, if you do not specify a base restructuring table, IDMSRSTT is used.

- **rstt-modname**
  The name of the base restructuring table. The default base restructuring table is IDMSRSTT.
AREA
Restricts processing to one or more specified areas.
By default, if you do not specify one or more areas, all areas in the specified segment that contain occurrences of the records or members of the sets being restructured are processed.

area-name
The name of an area.

Usage

How to submit the RESTRUCTURE CONNECT statement
You submit the RESTRUCTURE CONNECT statement only through the batch command facility. You must be running CA IDMS/DB in local mode.

Vary areas offline
Before submitting the RESTRUCTURE CONNECT statement, vary all affected areas offline.

Connecting pointers for new sets
The RESTRUCTURE CONNECT statement connects new pointers for existing sets only. To connect pointers for new sets, you must run a user-written program after the restructure process is complete.

JCL Considerations

When you submit a RESTRUCTURE CONNECT statement through the batch command facility, the JCL to execute the facility must include statements to define:

- The database file(s) that map to the area(s) to be processed.
- The file containing the assembled base restructuring table.
- The spill file generated by RESTRUCTURE. The size of the spill file should be a multiple of 40 with a maximum block size of 32,760.
- Sort files to sort the spill file in database key sequence.

Note: For more information about the generic JCL used to execute the batch command facility, see the Batch Command Facility (z/OS) (see page 355) or z/VSE JCL (see page 389) topics.
Examples and Sample Output

To illustrate the use of the RESTRUCTURE CONNECT utility, the RESTRUCTURE utility is run first to add pointers to a set. The RESTRUCTURE CONNECT utility is then executed to update the prefix portion of affected records with pointers.

Adding set pointers

The following example directs the RESTRUCTURE utility to use the base restructuring table, LRDKRSTT, to add prior pointers to the owners and members of set A-B and owner pointers to the members of set A-B.

**Input to RESTRUCTURE**

```
RESTRUCTURE empdemo using restr01
        rsttmod lrdkrstt continue yes;
```

**Output from RESTRUCTURE**

When the RESTRUCTURE operation is completed, the following listing is generated.

```
UT000038 Starting Restructure of area EMPDEMO.EMP-DEMO-REGION
UT000041 Completed processing of area EMPDEMO.EMP-DEMO-REGION, Pages read=10  Records read=6
UT011023  Record name AA  Found=2  Changed=2  Ldel=0
UT000038 Starting Restructure of area EMPDEMO.INS-DEMO-REGION
UT000041 Completed processing of area EMPDEMO.INS-DEMO-REGION, Pages read=10  Records read=3
UT011023  Record name BB  Found=3  Changed=3  Ldel=0
Status = 0
```

Connecting prior and owner pointers

To complete this restructure operation, the RESTRUCTURE CONNECT utility is executed to connect the prior and owner pointers in all occurrences of set A-B after restructuring records A and B.

**Input to RESTRUCTURE CONNECT**

```
restructure connect segment empdemo using EMPSS01
        rsttmod lrdkrstt continue yes;
```

**Output from RESTRUCTURE CONNECT**

When the RESTRUCTURE CONNECT operation is completed, the following listing is generated.

```
UT000038 Starting Connect of area EMPDEMO.EMP-DEMO-REGION
UT000041 Completed processing of area EMPDEMO.EMP-DEMO-REGION, Pages read=10  Records read=6
UT011023  Record name AA  Found=2  Changed=2  Ldel=0
```

16-Jan-2018 251/493
UT000041 Completed processing of area EMPDEMO.INSDEMO-REGION, Pages read=10 Records read=3
UT011023 Record name BB Found=3 Changed=3 Ldel=0
Status = 0

More Information

- For more information about changing database components, see Administrating CA IDMS Database (https://docops.ca.com/display/IDMS/Administrating+CA+IDMS+Database).
- For more information about the RESTRUCTURE utility, see RESTRUCTURE (see page 241).

ROLLBACK Utility

The ROLLBACK utility restores all or part of a database to a previous state by applying before images from the journal file.

If requested, the ROLLBACK utility verifies the after images in the journal file against the contents of the database before applying the before images.

This article describes the following information:

- Authorization (see page 252)
- Syntax (see page 252)
- Parameters (see page 253)
- Usage (see page 256)
- JCL Considerations (see page 259)
- Examples (see page 259)
- Sample Output (see page 259)

Authorization

<table>
<thead>
<tr>
<th>To</th>
<th>You Need This Privilege</th>
<th>On</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roll back an area</td>
<td>DBAWRITE</td>
<td>The area</td>
</tr>
<tr>
<td>Roll back a segment</td>
<td>DBAWRITE</td>
<td>All affected areas of the segment</td>
</tr>
<tr>
<td>Roll back a file</td>
<td>DBAWRITE</td>
<td>All affected areas associated with the file</td>
</tr>
<tr>
<td>Roll back a database</td>
<td>DBAWRITE</td>
<td>All affected areas of the database</td>
</tr>
</tbody>
</table>

Syntax

```
ROLLBACK
FILE segment-name.file-name
```
Parameters

- **FILE**
  Rolls back one or more files. Recovery by file does not unlock any areas associated with the files.

- **segment-name**
  Specifies the segment associated with the file.

- **file-name**
  Specifies the name of the file.

- **AREA**
  Rolls back and unlocks one or more specified areas.

- **segment-name**
  Identifies the segment associated with the area.

- **area-name**
  Specifies the name of the area.

- **SEGMENT**
  Rolls back and unlocks all areas associated with the specified segments.

- **segment-name**
  Specifies the name of the segment.

- **DMCL**
  Rolls back all areas defined in the DMCL identified in the SYSIDMS parameter file.

- **ACTIVE**
If you do not specify a STOP time, before images are applied only for transactions that remain open at the end of the journal file. Before images are not applied for incomplete distributed transactions whose state is InDoubt at the end of the journal file (unless a section recovery control file entry overrides this behavior by specifying BACKOUT for one or more of the transactions).

If you do specify a STOP time, before images are applied for all transactions until the STOP time is reached, and then processing continues only with open transactions. Before images are not applied for incomplete distributed transactions whose state is InDoubt at the stop time (unless a section recovery control file entry overrides this behavior by specifying BACKOUT for one or more of the transactions).

- **ALL**
  - If you do not specify a STOP time, before images are applied for all transactions in the journal file.
  - If you do specify a STOP time, before images are applied for all transactions until the STOP time is reached, and then processing continues only with open transactions.

- **SEQUENTIAL**
  Applies before images in the sequence in which they occur starting from the end of the journal file, reading backwards. SEQUENTIAL is the default.

- **SORTED**
  Sorts the before images in the journal file by database key and copies back only the earliest before image for any database key.
  All the database keys on a given database page will be copied back together. Each page will be accessed once.
  By default, if you do not specify SORTED, the before images will be copied back sequentially.

  **Note:** When recovering VSAM KSDS files, do not use the SORTED option. You must use SEQUENTIAL.

- **VERIFY DATABASE**
  Verifies the after images in the journal file before applying the before images. If an after image in the journal file does not match the contents of the database, the roll back operation will terminate with an error.
  By default, if you do not specify VERIFY, before images are copied back without first verifying the after images.
  The optional DATABASE keyword is included for consistency with the ROLLFORWARD utility statement and has no impact on the processing of the VERIFY option.

- **STOP AT**
  Directs processing to stop as soon as possible after reaching a specified date and time in the journal file.
  By default, processing will stop when the beginning of the journal file is reached.
• **date**
  Specifies the date, in one of the following formats:

  - `yyyy-mm-dd`
  - `mm/dd/yyyy`
  - `dd.mm.yyyy`

  In these formats, the following rules apply:

  - `yyyy` specifies the year. `yyyy` must be an integer in the range 0001 through 9999. Leading zeros are optional.
  - `mm` specifies the month within the year. `mm` must be an integer in the range 01 through 12. Leading zeros are optional.
  - `dd` specifies the day within the month. `dd` must be an integer in the range 01 through 31. Leading zeros are optional.

  The combined values of `yyyy`, `mm`, and `dd` must represent a valid date. For example, 1988-02-29 is a valid date. 1989-02-29 is not.

• **date-time**
  Specifies the date and time, where:

  - The format for specifying the DATE-TIME is: `yyyy-mm-dd-hh.mm.ss.fffffff`
  - The rules for specifying the DATE component of DATE-TIME are the same as for DATE described previously. The rules for specifying the TIME component of DATE-TIME are:

    - `hh` specifies the hour on a 24-hour clock. `hh` must be an integer in the range 00 through 23. Leading zeros are optional.
    - `mm` specifies the number of minutes past the hour. `mm` must be an integer in the range 00 through 59. Leading zeros are optional.
    - `SS` specifies the number of seconds past the minute. `SS` must be an integer in the range 00 through 59. Leading zeros are optional.
    - `fffffff` specifies the number of millionths of a second past the specified second. `fffffff` is optional; if you include it, it must be an integer in the range 000000 through 999999. The default value is 000000. Trailing zeros are optional.

• **time**
  Specifies the time in one of the following formats:

  - `hh.mm.ss`
  - `hh:mm:ss`
The rules for specifying TIME are the same as those listed for DATE-TIME. When TIME is specified, the date defaults to the current date.

- **HOTBACKUP**
  Forces ROLLBACK to also restore images for aborted run units.

- **UTC**
  Specifies that Start and Stop times are interpreted as UTC times instead of local times.

## Usage

### How to submit the ROLLBACK statement

You submit the ROLLBACK statement only through the batch command facility. When submitting ROLLBACK statements, you must run the batch command facility in local mode.

Vary any area being rolled back offline to all DC/UCF systems.

### When to use ROLLBACK

The ROLLBACK utility is most commonly used after a local mode update job has abnormally terminated or has been run incorrectly, resulting in incorrect data being introduced into the database. In this case, use the ROLLBACK utility to restore the affected parts of the database to an earlier, uncorrupted, state.

Both the database and the journal must be available and readable to run the ROLLBACK utility.

### When not to use ROLLBACK

If the database is unusable or incomplete because of a physical I/O error, then you cannot use the ROLLBACK utility.

In this case, use a backup copy of the database and run ROLLFORWARD.

> **Note:** For more information about when and how to use the ROLLBACK and ROLLFORWARD utilities, see [Administrating CA IDMS Database](https://docops.ca.com/display/IDMS/Administrating+CA+IDMS+Database).

### How ROLLBACK works

The ROLLBACK utility starts with the current state of the database. It uses before images from a tape or journal file to restore progressively earlier states of the database, file, segment, or area.

You can run the ROLLBACK utility **sequentially**, applying before images in reverse order. Or you can run the ROLLBACK utility in **sorted** mode. In sorted mode, the ROLLBACK utility applies the earliest before image of any record being restored.

**ROLLBACK uses one journal file**
The journal file is a sequential file that can reside on disk or tape. If you have multiple journal files, you must consolidate them, in the order in which they were created, to a single file and use the single file with the ROLLBACK utility.

**Disk journal file**

If the journal file is on disk, it cannot be read backwards in certain operating systems. To accommodate this, a SYSIDMS PARM has been created, ROLLBACK3490. To utilize this feature, the journal file must be sorted on the first UTC timestamp of each journal block, in a descending order. In this manner, the ROLLBACK utility, using the ROLLBACK3490 feature, reads each block forward, but is actually getting the blocks in descending order. Although the blocks are in descending order, the journal records within the blocks are in ascending order. As ROLLBACK reads each block in descending order, it processes each journal block from the end to the beginning. So even though the journal file is a sequential file on disk, sorted descending, ROLLBACK is processing the journal records in a descending order. To sort the journal blocks in descending order, use the following sort parm:

```
SORT FIELDS=(13,16,BI,D)
```

⚠️ **Note:**

- In CMS, you do not need to use the ROLLBACK3490 parameter, but you must presort the journal images as described previously regardless of whether the journal files reside on disk or tape.
- If you decide to use the ROLLBACK3490 feature, or are using CMS, then you must sort the file in descending order prior to running the ROLLBACK utility, even if you are using 'sorted mode.'

**Tape journal file**

If you want the journal file on tape, you can put the file on tape by using the ARCHIVE JOURNAL utility, or by using a tape journal to begin with. If you have multiple tape journal files, you must concatenate them, in the order in which they were created, to a single file; and use the single file with the ROLLBACK utility.

If the tape file requires more than one tape volume, the volumes must be mounted in the order in which they were created.

⚠️ **Note:** Depending on the version of z/OS that you are using and whether the tape volumes are labeled, you might need to mount either the first tape volume produced or all tape volumes produced before they can be read backwards.

**ROLLBACK and VSAM files**
You can run the ROLLBACK utility against most native VSAM files. However, because VSAM does not support deleting ESDS file records, the ROLLBACK statement cannot be used on a native VSAM ESDS file that might cause the ROLLBACK utility to try to delete a record. Therefore, a run unit that has added records to an ESDS file can only be recovered by restoring the file to an earlier time and then executing the ROLLFORWARD utility.

**Note:** For more information about CA IDMS/DB and native VSAM files, see [Administrating CA IDMS Database](https://docops.ca.com/display/IDMS/Administrating+CA+IDMS+Database).

**When to use sequential mode**

Use sequential mode if the journal file is very small or very large. If the journal file is small, you get no advantage from sorting. If the journal file is very large, it could require more sort space than you have available.

**When to use sorted mode**

Use sorted mode unless you have a strong reason not to. Be sure you have enough sorting space available. If there is not enough sort space available during ROLLBACK processing, the operation terminates with an error. The database will not be affected.

**When to use VERIFY**

Specify VERIFY only when rolling back an intact database. If you specify VERIFY when recovering from an abended job or an abended central version, the ROLLBACK utility terminates with an error.

VERIFY is ignored if you specify SORTED.

**Recovering large numbers of files**

For z/OS users, the maximum number of files that can be accessed by a central version is greater than the number that can be accessed by a local mode batch job. This has implications for section recovery. If more than 3,273 files must be recovered, it will be necessary to execute the ROLLBACK utility multiple times in separate job steps, recovering a subset of the areas or segments in each execution.

**When to use HOTBACKUP**

Specify HOTBACKUP when restoring a database from a hot backup. For more information about hot backup, see the CA IDMS Database Administration section.

**ROLLBACK and Distributed Transactions**

ROLLBACK reports on distributed transactions and supports the use of an input section recovery control file. The input recovery control file is used to complete InDoubt distributed transactions. For more information, see [JCL Considerations](https://docops.ca.com/display/IDMSCALL19/ROLLBACK#ROLLBACK-JCLConsiderations) and the “Common Facilities for Distributed Transactions” section.
JCL Considerations

When you submit a ROLLBACK statement through the batch command facility, the JCL to execute the facility must include statements to define:

- The tape or journal file
- All files that map to areas being restored
- Any file that includes space management pages of areas mapped to by files being restored

To use a section recovery input control file, include a CTRLIN file definition or DD statement in the IDMSBCF execution JCL. The format of this file is fixed block with a record length of 80.

Note: For more information about the generic JCL used to execute the batch command facility, see the section for your operating system in this section.

Examples

The following example directs the ROLLBACK utility to apply all the before images on the journal file for the EMPDEMO segment and to sort the before images by database key.

    rollback segment empdemo all sorted;

Sample Output

When the ROLLBACK operation is completed, the following listing is generated.

        IDMSBCF
        mm/dd/yy PAGE 1        IDMS Batch Command Facility
   ROLLBACK SEGMENT EMPDEMO ALL SORTED;

   RECORDS RESTORED TO AREA EMPDEMO.EMP-DEMO-REGION 734
   RECORDS RESTORED TO AREA EMPDEMO.INS-DEMO-REGION 169
   RECORDS RESTORED TO AREA EMPDEMO.ORG-DEMO-REGION 438
   TOTAL RECORDS RESTORED 1341
   BLOCK COUNT 0 BACKWARD 782
   RECORD COUNT 0 BACKWARD 5512
   Status = 1 Extended Reason Code = 2367 Messages follow:
DB002367 C1M353: Warning: area EMPDEMO.EMP-DEMO-REGION was not locked.
DB002367 C1M353: Warning: area EMPDEMO.INS-DEMO-REGION was not locked.
DB002367 C1M353: Warning: area EMPDEMO.ORG-DEMO-REGION was not locked.

More Information

- For more information about journal files, see Administrating CA IDMS Database (https://docops.ca.com/display/IDMS/Administrating+CA+IDMS+Database).
- For more information about database recovery, Administrating CA IDMS Database (https://docops.ca.com/display/IDMS/Administrating+CA+IDMS+Database).
- For more information about using DBNAME for utility use, see Administrating CA IDMS Database (https://docops.ca.com/display/IDMS/Administrating+CA+IDMS+Database).

**ROLLFORWARD**

The ROLLFORWARD utility restores a backup copy of all or part of a database to a subsequent condition by applying after images from the journal file.

If requested, the ROLLFORWARD statement verifies the before images in the journal file against the contents of the database before applying the after images.

This article describes the following information:

- Authorization (see page 260)
- Syntax (see page 261)
- Parameters (see page 261)
- Usage (see page 264)
- JCL Considerations (see page 268)
- Examples (see page 268)
- Sample Output (see page 268)

**Authorization**

<table>
<thead>
<tr>
<th>To</th>
<th>You Need This Privilege On</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roll forward an area</td>
<td>DBAWRITE The area</td>
</tr>
<tr>
<td>Roll forward a segment</td>
<td>DBAWRITE All affected areas of the segment</td>
</tr>
<tr>
<td>Roll forward a file</td>
<td>DBAWRITE All affected areas that are associated with the file</td>
</tr>
<tr>
<td>Roll forward by DMCL</td>
<td>DBAWRITE All affected areas within the DMCL</td>
</tr>
<tr>
<td>Roll forward FROM EXTRACT</td>
<td>DBAWRITE All areas being recovered</td>
</tr>
</tbody>
</table>
Syntax

ROLLFORWARD

FILE segment-name.file-name
AREA segment-name.area-name
SEGMENT segment-name
DMCL

rollforward-options
FROM EXTRACT ddname

Expansion of rollforward-options
ALL

SEQUENTIAL
SORTED

VERIFY
DATABASE
QUIESCE
NONE
LIMITED
FULL

UTC

START AT date-time
date
time

STOP AT date-time
date
time

Parameters

- **FILE**
  Specifies that one or more files is to be restored. Areas that are associated with the specified files are not unlocked.

- **segment-name**
  Specifies the segment that is associated with the file.

- **file-name**
  Specifies the name of the file.

- **AREA**
  Specifies that one or more areas are to be restored and unlocked.

- **segment-name**
  Identifies the segment that is associated with the area.

- **area-name**
  Specifies the name of the area.
• SEGMENT
  Specifies to restore and unlock all areas that are associated with the specified segments.

• segment-name
  Specifies the name of the segment.

• DMCL
  Restores all areas that are defined by the DMCL specified in the SYSIDMS parameter file.

• rollforward-options
  Specifies options that are used to rollforward from a standard archive journal tape.
  Expansion of rollforward-options immediately follows the main statement syntax.

• FROM EXTRACT
  Specifies that an extract journal is used for input instead of a standard archive journal tape.
  This option excludes use of the SORTED/SEQUENTIAL, VERIFY, ALL/COMPLETED, and STOP AT
  options because these options have no effect on the already created extract file.

• ddname
  Specifies the external name for the input extract file.

• COMPLETED

  ▪ If you do not specify a STOP time, after images are applied only for transactions that are
    completed before the end of the journal file. After images are applied for incomplete
    distributed transactions whose state is InDoubt at the end of the journal file (unless a section
    recovery control file entry overrides this behavior by specifying BACKOUT for one or more of
    the transactions).

  ▪ If you do specify a STOP time, after images are applied for all transactions that have been
    completed up to the first checkpoint record after the STOP time. After images are applied for
    incomplete distributed transactions whose state is InDoubt at the stop time (unless a section
    recovery control file entry overrides this behavior by specifying BACKOUT for one or more of
    the transactions).

• ALL

  ▪ If you do not specify a STOP time, after images are applied for all transactions in the journal
    file.

  ▪ If you do specify a STOP time, after images are applied until a point at or after the specified
    time where there are no active transactions. If no such point is found, processing stops at the
    end of the journal file.

• SEQUENTIAL
  After images are applied in the sequence in which they occur in the journal file.
  SEQUENTIAL is the default.

⚠️ Note: Backward file reads are not supported under z/VM. If an aborted transaction is
encountered, a backward read may be attempted, in which case it fails. For this reason, it
is safer for z/VM users to specify SORTED.
• **SORTED**
  Specifies that after images are to be sorted in the journal file by database key and only the latest after image for any database key is applied.
  All the database keys on a given database page are copied back together. Thus, each page is accessed once.
  By default, if you do not specify SORTED, the after images are copied back sequentially.

  **Note:** You cannot use the SORTED option with VSAM KSDS files. A native VSAM KSDS file must be recovered using the SEQUENTIAL option.

• **VERIFY**
  Indicates that the validity of the specified conditions should be checked. If the VERIFY option is not specified, checking is not performed. If VERIFY is specified without DATABASE or QUIESCE, DATABASE is the default.

  • DATABASE -- Specifies that before images are to be verified in the journal file before the after images are applied. If a before image in the journal file does not match the contents of the database, the rollforward operation terminates with an error.
    By default, if database validity checking is not in effect, after images are copied back without first verifying the before images.

  • QUIESCE -- Specifies the level of quiesce point verification that is performed.
    • NONE -- Specifies that no quiesce point verification should be done. If "START AT" is specified, NONE is treated as LIMITED.
    • LIMITED -- Default. Specifies that limited quiesce point verification be performed.
    • FULL -- Specifies that the strictest form of quiesce point verification be performed.

• **START AT**
  Specifies that the operation should start only after reaching a specified date and time in the journal file. Only images for transactions that begin after the specified start time is applied. By default, processing starts at the beginning of the journal file.

• **STOP AT**
  Specifies that the operation should stop as soon as possible after reaching a specified date and time in the journal file.
  By default, processing stops when the end of the journal file is reached.

• **date**
  Specifies the date, in one of the following formats:

  • yyyy-mm-dd
  • mm/dd/yyyy
  • dd.mm.yyyy

  In these formats, the following rules apply:
• **yyyy** specifies the year. **yyyy** must be an integer in the range 0001 through 9999. Leading zeros are optional.
  
  • **mm** specifies the month within the year. **mm** must be an integer in the range 01 through 12. Leading zeros are optional.

• **dd** specifies the day within the month. **dd** must be an integer in the range 01 through 31. Leading zeros are optional.

The combined values of **yyyy**, **mm**, and **dd** must represent a valid date. For example, 1988-02-29 is a valid date. 1989-02-29 is not.

• **date-time**

  Specifies the date and time, where:

  • The format for specifying the DATE-TIME is:
    
    **yyyy-mm-dd-hh.mm.ss.ffffff**

  • The rules for specifying the DATE component of DATE-TIME are the same as for DATE described previously. The rules for specifying the TIME component of DATE-TIME are:

    • **hh** specifies the hour on a 24-hour clock. **hh** must be an integer in the range 00 through 23. Leading zeros are optional.

    • **mm** specifies the number of minutes past the hour. **mm** must be an integer in the range 00 through 59. Leading zeros are optional.

    • **ss** specifies the number of seconds past the minute. **ss** must be an integer in the range 00 through 59. Leading zeros are optional.

    • **ffffff** specifies the number of millionths of a second past the specified second. **ffffff** is optional; if you include it, it must be an integer in the range 000000 through 999999. The default value is 000000. Trailing zeros are optional.

• **time**

  Specifies the time in one of the following formats:

  • **hh.mm.ss**

  • **hh:mm:ss**

  The rules for specifying TIME are the same as listed for DATE-TIME. When TIME is specified, the date defaults to the current date.

• **UTC**

  Specifies that Start and Stop times are interpreted as UTC times instead of local times.
You submit the ROLLFORWARD statement only through the batch command facility. When submitting ROLLFORWARD statements, you must run the batch command facility in local mode.

Vary any area being rolled forward offline to all central versions.

**When to use ROLLFORWARD**

The ROLLFORWARD utility is most commonly used to restore a database to its condition before some hardware problem corrupted the database. In this case, use a backup copy of the database that was created before the problem occurred.

You must have a backup of the database from a time before the problem, and all relevant journals.

**How ROLLFORWARD works**

The ROLLFORWARD operation starts with a former state of the database, as captured by a backup. ROLLFORWARD uses after images from a tape or journal file to restore progressively later states of the database, file, segment, or area.

**ROLLFORWARD uses one journal file**

The journal file is a sequential file that can reside on disk or tape depending on which options are specified. If you specify SEQUENTIAL, the journal file must reside on tape. If you specify SORTED, the journal file can reside on disk or tape. If you have multiple journal files, you must consolidate them, in the order in which they were created, to a single file, and use the single file with the ROLLFORWARD utility.

If the file requires more than one tape volume, the volumes must be mounted in order.

**ROLLFORWARD and VSAM files**

You can run the ROLLFORWARD utility against native VSAM files.

**When to use sorted mode**

Use sorted mode unless you have a strong reason not to. Be sure that you have enough sort space available before executing the ROLLFORWARD utility. If, however, you run out of sort space, the ROLLFORWARD operation terminates with an error, and the database is not affected.

**ROLLFORWARD FROM EXTRACT**

- **Purpose** -- Much of the work of recovery can be performed by EXTRACT JOURNAL during a noncritical time. The actual recovery that is done by ROLLFORWARD is fast because the extract file contains only the records that are needed for recovery, in correct sort order.

- **Extract data** -- Data in the extract file is selected and sorted before the actual recovery is done by ROLLFORWARD. A second sort is performed by ROLLFORWARD so that only the most recent image for each db-key is applied to the database.

- **Multiple input files** -- Multiple input files can be processed as standard concatenated files; they do not need to be merged into one file before processing.
- **Tape file** -- Because the extract file does not need to be read backwards, it does not need to be on tape.

**Controlling the starting point of the rollforward operation**

If no START AT parameter is specified, the rollforward operation starts with the first journal image on the input file(s). If START AT is specified, the rollforward operation starts with the first checkpoint record (BGIN, COMT, ENDJ, ABRT, or CKPT) whose timestamp is on or after the specified start time.

Specifying a start time may save recovery time and also circumvent issues that are associated with aborted recovery units that span the start of the input file. It further enables the rollforward operation to begin processing at a quiesce point that does not correspond with a journal file boundary.

**Quiesce point verification**

A quiesce point is a point in time during which there is no update activity for some portion of a database. To perform a correct recovery, you must begin the recovery operation at a quiesce point for the portion of the database being recovered. To assist in this effort, the rollforward utility provides two levels of quiesce point verification: limited and full.

Quiesce point verification is always performed during a rollforward operation. Verification is done by checking for the existence of spanned recovery units, which are recovery units that are active at the start of the rollforward operation. A recovery unit is represented by journal images starting with a BGIN or COMT checkpoint and ending with a COMT, ENDJ, or ABRT checkpoint. If a spanned recovery unit updates the specified portion of the database, then the rollforward operation is not starting at a quiesce point. If this situation is detected, the rollforward operation terminates with an error.

However, it is not always possible to know whether a spanned recovery unit affects the specified portion of the database or not. If the initial BGIN or COMT checkpoint record for a recovery unit is not contained on the archive file being processed, then it is not possible to determine whether it updated the specified portion of the database. Such a recovery unit is referred to as an indoubt recovery unit.

The following time line example illustrates what is meant by an indoubt recovery unit. The journal images for recovery unit R are written to the journal file at the times shown. If the archive file includes images starting at time T1, then R is not an indoubt recovery unit because the archive file contains all journal images that are written for R since its inception. Similarly, if the archive file starts at time T3, R is not an indoubt recovery unit, because the archive file contains no images for R whatsoever. However, if the archive file starts at time T2, then R is an indoubt recovery unit, because the archive file does not contain all journal images that are written since its inception.
If an indoubt recovery unit does not span the start of the rollforward operation, its existence does not matter. But if an indoubt recovery unit is also a spanned recovery unit, then the rollforward operation may not be starting at a quiesce point.

The action that is taken if an indoubt spanned recovery unit is encountered depends on whether limited or full quiesce point verification is in effect. Under full verification, the rollforward operation terminates with an error. Under limited verification, a warning message is issued identifying the recovery unit, but processing continues.

Warning messages that are produced under limited verification should be examined to ensure that the identified recovery units in fact did not affect the specified portion of the database. If there is any doubt, the PRINT JOURNAL utility statement should be used to gain more information about the indoubt recovery units. If, after researching the situation, it is found that an indoubt recovery unit did update the specified portion of the database, the affected portion of the database must be restored, and the rollforward operation repeated. You must locate a quiesce point corresponding to a backup of the specified portion of the database and begin the rollforward operation from that point.

**When to use full or limited verification**

Full quiesce point verification should be used only if you expect that no indoubt spanned recovery units are active at the starting point of the rollforward operation. The only way to ensure this is to process archive files that were created immediately following a quiesce of update activity across all areas. One way to establish such a quiesce point is to shut down a central version. Another way is to use the DCMT QUIESCE command and specify a DBNAME that includes every area in the DMCL.

Limited quiesce point verification can be used when processing the archive files produced immediately following a quiesce operation for the portion of the database being recovered. One way to do this is to use the DCMT QUIESCE SEGMENT command to quiesce a segment and then use limited quiesce point verification when recovering all or a portion of that segment.

**When to use VERIFY DATABASE**

Specify VERIFY DATABASE only when rolling forward from an intact database.

**Recovering large numbers of files**

For z/OS users, the maximum number of files that can be accessed by a central version is greater than the number that can be accessed by a local mode batch job. This has implications for section recovery. If more than 3,273 files must be recovered, it is necessary to execute the ROLLFORWARD utility multiple times in separate job steps, recovering a subset of the areas or segments in each execution.

**ROLLFORWARD and Distributed Transactions**

ROLLFORWARD reports on distributed transactions and supports the use of input and output section recovery control files. Unless ALL is specified, the input section recovery control file is used to complete InDoubt distributed transactions. If an output section recovery control file is included in the JCL, unless FROM EXTRACT is specified, an entry is written for each incomplete distributed transaction. For more information, see JCL Considerations (see page 268) and the "Common Facilities for Distributed Transactions" section.
JCL Considerations

When you submit a ROLLFORWARD statement through the batch command facility, the JCL to execute the facility must include statements to define:

- The tape or journal file or journal extract file
- All files that map to areas being restored
- Any file that includes space management pages of areas that are mapped to by files being restored
- Sort work and message files if the sort option is selected or if recovering from a journal extract file

To use a section recovery input control file, include a CTRLIN file definition or DD statement in the IDMSBCF execution JCL. To use a section recovery output control file, include a CTRLOUT file definition or DD statement in the IDMSBCF execution JCL. The format of both of these files is fixed block with a record length of 80.

⚠️ **Note:** For more information about the generic JCL used to execute the batch command facility, see the section for your operating system in this section.

Examples

The following example directs the ROLLFORWARD utility to restore the EMPDEMO segment using the SORTED option.

```bash
rollforward segment empdemo all sorted;
```

The following example directs the ROLLFORWARD utility to read an extract file from SYS005 and apply only the images that belong to file USERDB.EMPF1:

```bash
rollforward file userdb.empf1 from extract sys005;
```

Sample Output

The ROLLFORWARD utility generates the following listing after executing the statement in the first example.

```
IDMSBCF mm/dd/yy PAGE 1

ROLLFORWARD SEGMENT EMPDEMO ALL SORTED;

RECORDS RESTORED TO AREA EMPDEMO.EMP-DEMO-REGION 1,030
```
RECORDS RESTORED TO AREA EMPDEMO.INS-DEMO-REGION 199
RECORDS RESTORED TO AREA EMPDEMO.ORG-DEMO-REGION 581
TOTAL RECORDS RESTORED 1810

BLOCK COUNT 702  BACKWARD  0
RECORD COUNT 5512  BACKWARD  0
Status = 1  Extended Reason Code = 2367  Messages follow:
DB002367 C1M353: Warning: area EMPDEMO.EMP-DEMO-REGION was not locked.
DB002367 C1M353: Warning: area EMPDEMO.INS-DEMO-REGION was not locked.
DB002367 C1M353: Warning: area EMPDEMO.ORG-DEMO-REGION was not locked.

More Information

For more information, see Administrating CA IDMS Database. (https://docops.ca.com/display/IDMS19/Administrating+CA+IDMS+Database)

SYNCHRONIZE STAMPS

The SYNCHRONIZE STAMPS utility manipulates SQL synchronization stamps in the following ways:

- Displays and compares the stamps in the catalog and the data area(s), issuing a warning if stamps are inconsistent
- Updates the catalog with the stamps from the data area(s)
- Updates the data area(s) with the stamps from the catalog

This utility provides an alternate mechanism for taking snapshot copies of identically defined databases and also as an aid in recovery situations in which either the catalog or a data area must be restored independently of each other.

This article describes the following information:

- Syntax (see page 269)
- Parameters (see page 270)
- Usage (see page 270)
- JCL Considerations (see page 271)
- Examples (see page 271)
- Sample Output (see page 271)

Authorization

<table>
<thead>
<tr>
<th>To</th>
<th>You Need This Privilege</th>
<th>On</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use SYNCHRONIZE STAMPS</td>
<td>DBAWRITE</td>
<td>Every area processed by SYNCHRONIZE STAMPS</td>
</tr>
</tbody>
</table>

Syntax

SYNCHRONIZE STAMPS FOR AREA segment-name.area-name

16-Jan-2018  269/493
Parameters

- **AREA**
  Specifies the area in which to synchronize stamps.

- **segment-name**
  Specifies the name of the segment associated with the area.

- **area-name**
  Specifies the name of an area included in the DMCL module.

- **SEGMENT**
  Specifies the segment whose areas will have their stamps synchronized.

- **segment-name**
  Specifies the name of a segment included in the DMCL module.

- **UPDATE CATALOG**
  Specifies that the catalog is to be updated with the stamps from the data area(s).

- **UPDATE DATABASE**
  Specifies that the area(s) are to be updated with stamps from the catalog.

- **COMPARE**
  Displays and compares the stamps in the catalog and the data area(s) and issues a warning if the stamps are inconsistent.

Usage

**How to submit the SYNCHRONIZE STAMPS statement**

You submit the SYNCHRONIZE STAMPS statement using either the batch command facility or the online command facility.

**Use caution when updating stamps**

By using the SYNCHRONIZE STAMPS utility to update stamps in a catalog or data area, you are asserting that the definition in the catalog accurately describes data in the area. You should be sure that this is true before updating stamp values. No data validation is performed by the utility.
JCL Considerations

When you submit a SYNCHRONIZE STAMPS statement through the batch command facility in local mode, the JCL to execute the facility must include statements to define the following:

- File(s) that map to the area to be processed
- Dictionary that defines the table(s) in the area
- Journal file(s) if backups are not taken

To run under central version, a SYSCTL statement is needed.

⚠️ Note: For more information about the generic JCL used to execute the batch command facility, see the section for your operating system in this section.

Examples

The following example directs the SYNCHRONIZE STAMPS utility to compare the synchronization stamps in the USERDB.EMP_AREA with those in the catalog.

```
synchronize stamps for area userdb.emp_area compare;
```

Sample Output

The following report sample is produced by the SYNCHRONIZE STAMPS utility.

```
IDMSBCF   nn.n    CA IDMS Batch Command Facility mm
/dd/yy    PAGE 1
SYNCHRONIZE STAMPS FOR AREA USERDB.EMP_AREA COMPARE;
** Status = 0 SQLSTATE = 00000 *** Current Stamps Report ***
Area:USERDB.EMP_AREA Table Stamping
Catalog Stamp: <null>

Database Stamp: <null>
```

Table ID:1024 Table:DEMO.EMPLOYEE
Catalog Stamp: 1993-03-08-14.50.01.952955
Database Stamp: 1993-03-08-14.50.01.952955

Table ID:1025 Table:DEMO.POSITION
Catalog Stamp: 1993-03-08-14.50.01.668147
Database Stamp: 1993-03-08-14.50.01.668147

Table ID:1026 Table:DEMO.MANAGERS
Catalog Stamp: 1993-03-08-14.50.01.952955
Database Stamp: 1993-03-08-14.50.01.952955

Table ID:1027 Table:INV.PART
Catalog Stamp: 1996-06-18-10.40.51.839925
```
TUNE INDEX

The TUNE INDEX utility performs the following functions:

- Adopts orphans in an index structure. An orphaned indexed record is a record whose index pointer does not point back to the index record (SR8) that contains the record's index entry. Orphans occur as the result of splitting an existing SR8 into two records to accommodate a new entry. As part of the split, some of the entries are moved to a new SR8, but the index pointer in their associated records is not adjusted to reflect the change, resulting in "orphaned" records. By eliminating orphans, runtime database performance is improved when traversing from an indexed record to its associated index entry.

- Moves the top level SR8 to its optimal location.

- Optionally rebalances the index structure. Rebalancing ensures that the resulting index structure is a balanced tree and has a minimal number of levels and SR8's. You can temporarily override the index block contains value of the index and the page reserve value of the area that contains the index structure. Using these overrides allows tuning the index while allowing for future growth.

- Optionally resequences the index structure. Resequencing puts the SR8 records in physical sequence. By resequencing the index structure, database performance is improved when accessing the index structure sequentially at the bottom level.

Note: Under some circumstances the process of moving the top level SR8 to its optimal location might cause some degree of resequencing to be performed even if RESEQUENCE NO is specified, or when the RESEQUENCE option is allowed to default to NO.
Authorization

A user must have DBAWRITE access on all areas processed by the utility.

Syntax

```
TUNE INDEX FOR
  AREA sql-area-name
  TABLE [schema-name.] table-name [ix-or-con-spec] [tune-options]
  DBNAME dbname SUBSCHEMA ss-name [set-spec]
  DEFAULT TUNE OPTIONS [tune-options]

COMMIT INTERVAL cmt-interval

NOTIFY INTERVAL not-interval
```

Expansion of `ix-or-con-spec`

```
CONSTRANIT constraint-name
  INDEX index-name
  (constraint-name [tune-options])
```

Expansion of `set-spec`

```
SET set-name
  (set-name [tune-options])
```

Expansion of `tune-options`

```
REBALANCE [NO YES]
RESEQUENCE [NO YES]
TEMPORARY INDEX UTILIZATION IS [ixutil-pct] PERCENT [key-count %]
```
Parameters

- **schema-name**
  Identifies the schema that will qualify the table name. If omitted, the current session associated with the user session schema is used. Schema-name is required if there is no current session schema.

- **table-name**
  Identifies the table that is constrained by an indexed constraint.

- **ix-or-con-spec**
  Identifies constraints and indexes on the current table that is being tuned. If omitted, all indexed constraints and indexes on the current table are processed.

- **sql-segment.area-name**
  Identifies an sql-defined area to be selected for processing. All tables with indexed constraints in the area are processed.

- **dbname**
  Identifies the dbname to be used when binding the subschema.

- **ss-name**
  Identifies the subschema to be used for processing a non-SQL database.

- **set-name**
  Identifies the indexed sets within the subschema that are to be processed. If omitted, all linked indexed sets defined in the subschema are processed. (Linked indexed sets are indexed sets with index pointers.)

- **cmt-interval**
  Specifies the commit interval. After every cmt-interval record read, a commit is issued. If omitted, the default interval is 100. If specified as zero (0), no commits are issued.

- **not-interval**
  Specifies the length of the notify interval in minutes. After each interval expires, a message is issued stating how far the job has progressed. If the notify interval is specified as 0 or allowed to default to 0, no notify messages are created.

- **constraint-name**
  Identifies an indexed constraint on the current table that is to be tuned.

- **index-name**
  Identifies an index on the current table that is to be tuned.

- **DEFAULT TUNE OPTIONS**
  The tune options to be used during the processing of an index if tune options have not been specifically specified.
• **REBALANCE**
  Specifies whether to rebalance the index. A well-balanced index has the minimum number of index levels and best performance if the index is frequently accessed vertically from top to bottom.

  - **YES**
    Rebalances the index.

    ![Warning: Rebalancing an index can be resource-intensive.]

  - **NO**
    No rebalancing is done.

• **RESEQUENCE**
  Specifies whether to resequence the index. A properly sequenced index is important only if the index is frequently accessed sequentially at the bottom level.

  - **YES**
    Resequences the index for optimum performance.

    ![Warning: Resequencing an index can be resource-intensive.]

  - **NO**
    No resequencing is done.

• **TEMPORARY INDEX UTILIZATION**
  Specifies a temporary override for the operation. If not specified, the current run-time value for INDEX BLOCK CONTAINS is used and index blocks are used at 100%.

  - **ixutil-pct**
    Specifies the percentage of the maximum number of entries that each index block should contain after tuning is complete. *ixutil-pct* is an integer in the range 10 through 100. The number of entries of an index block is computed as \( \text{index-block-contains} \times \frac{\text{ixutil-pct}}{100} \).

  - **key-count**
    Specifies the maximum number of entries that each index block should contain after tuning is complete. *key-count* is an integer in the range 3 through 8180.

    ![Warning: If the specified value exceeds the current run-time value of the INDEX BLOCK CONTAINS, the key-count value is ignored and the INDEX BLOCK CONTAINS value will be used.]

• **TEMPORARY PAGE RESERVE**
  Specifies a temporary override of the page reserve for the area in which the index resides. If not specified or specified as NULL, the page reserve of the area in which the index resides is used.
Usage

General Considerations

The TUNE INDEX utility has the following usage considerations:

- To use the TUNE INDEX utility, you must specify one of the following:
  - One or more tables whose indexed constraints are to be tuned
  - One or more areas containing tables whose indexed constraints are to be tuned
  - A subschema and DBNAME and optionally a list of indexed sets to be tuned

- If multiple indexes and/or multiple tables are processed in the same area, increasing the number of buffers further improves performance.

- Index tuning is a resource-intensive operation consisting mostly of CPU and I/O.

Operating modes

You can execute the TUNE INDEX utility both online (through the online command facility) and in batch through central version or batch local. When index tuning is executed by a central version, TUNE INDEX tries to minimize impact on other online tasks as follows:

- When a record or area lock conflict occurs with other applications, TUNE INDEX takes the following actions:
  - For record lock conflicts, TUNE INDEX commits the updates done so far.
  - For area lock conflicts, TUNE INDEX finishes its database session and starts a new one.

- TUNE INDEX lowers its priority to one below its normally assigned priority. If a deadlock occurs, the default deadlock selection algorithm selects the task with the lowest priority as the deadlock victim. The TUNE INDEX utility can recover from a deadlock by restarting the index tuning process of the current index occurrence.

Commit interval

You can specify a commit interval that determines the frequency with which the utility will commit. The interval specifies the number of updates that can take place before a commit is issued. You can disable committing and automatic restart by specifying a 0-commit interval. Regardless of the commit

page-reserve-pct

Specifies the percentage of each page to leave as free space if it contains a portion of an index being tuned. page-reserve-pct is an integer in the range 0 through 30. The page reserve of the area is computed as area-page-size * page-reserve-pct / 100.

reserve-character-count

Specifies the number of characters to reserve on each page to accommodate increases in the length of records or rows stored on the page if it contains a portion of an index being tuned. reserve-character-count is an integer with a value not larger than 30% of the page size.
interval specified, the utility always issues a COMMIT ALL at the end of the tune process of an index occurrence to release all record locks. It also issues a COMMIT ALL if it detects that another task is waiting on a record lock that it holds and it issues a FINISH if it detects that another task is waiting on an area lock that it holds.

**Notify interval**

You can specify a time interval in minutes. Each time this interval expires, a message is written indicating the index tuning progress. The message is written to the job log and the operator's console if TUNE INDEX runs in local mode; otherwise, it is written to the IDMS LOG and console. You can disable notification by specifying a 0-notify interval.

**TUNE OPTIONS Usage**

When processing multiple indexes within a single execution of the TUNE INDEX utility the tune options to be used are determined by the following hierarchy:

1. Options specified for a particular index.
2. Options specified on a DEFAULT TUNE OPTIONS statement
3. The utility default values.

**JCL Considerations**

When you submit a TUNE INDEX utility through the batch command facility in local mode, the JCL to execute the facility must include statements to define the files containing the areas to be processed. To run under central version, a SYSCTL statement is needed.

**Examples**

The following example directs the TUNE INDEX utility to adopt orphaned index records in the EMPDEMO dbname using subschema EMPSS01:

```
tune index for dbname empdemo subschema empss01;
```

The following example directs the TUNE INDEX utility to adopt orphaned index records, rebalance and resequence the index. It also shows how to temporarily override the DMCL or subschema values for PAGE RESERVE and INDEX BLOCK CONTAINS.

```
TUNE INDEX FOR DBNAME EMPDEMO SUBSCHEMA EMPSS01
  SET (EMP-NAME-NDX)
  DEFAULT TUNE OPTIONS
    REBALANCE YES
    RESEQUENCE YES
    TEMPORARY INDEX UTILIZATION IS 80 %
    TEMPORARY PAGE RESERVE IS 15 PERCENT
  NOTIFY INTERVAL 1000;
```
Sample Output

The following example directs the TUNE INDEX utility to adopt orphaned index records in the EMPDEMO dbname using subschema EMPSS01:

IDMSBCF  nn.n  CA IDMS Batch Command Facility
TUNE INDEX FOR DBNAME EMPDEMO SUBSCHEMA EMPSS01;
Status = 0  SQLSTATE = 00000  Messages follow:
DB002994 COM333: IDMSTUNE - processing started
DB002994 COM333: IDMSTUNE - Indexes selected for processing:
DB002994 COM333: IDMSTUNE - SKILL-EXPERTISE (IBC=30) in area EMPDEMO.ORG-DEMO-REGION (PGRSV=0)
DB002994 COM333: IDMSTUNE - EMP-NAME-NDX (IBC=40) in area EMPDEMO.EMP-DEMO-REGION (PGRSV=0)
DB002994 COM333: IDMSTUNE - OFFICE-EMPLOYEE (IBC=30) in area EMPDEMO.ORG-DEMO-REGION (PGRSV=0)
DB002994 COM333: IDMSTUNE - SKILL-NAME-NDX (IBC=30) in area EMPDEMO.ORG-DEMO-REGION (PGRSV=0)
DB002994 COM333: IDMSTUNE - JOB-TITLE-NDX (IBC=30) in area EMPDEMO.ORG-DEMO-REGION (PGRSV=0)
DB002994 COM333: IDMSTUNE - Statistics for area EMP-DEMO-REGION
DB002994 COM333: IDMSTUNE - Orphan adoption read 36 records (of which 8 SR8s)
DB002994 COM333: IDMSTUNE - Orphan adoption adopted 20 index orphans
DB002994 COM333: IDMSTUNE - Resequencing read 4 records
DB002994 COM333: IDMSTUNE - Statistics for area ORG-DEMO-REGION
DB002994 COM333: IDMSTUNE - Orphan adoption read 259 records (of which 63 SR8s)
DB002994 COM333: IDMSTUNE - Orphan adoption adopted 0 index orphans
DB002994 COM333: IDMSTUNE - Resequencing read 126 records
DB002994 COM333: IDMSTUNE - 425 total records read
DB002994 COM333: IDMSTUNE - 20 total index orphans adopted
DB002994 COM333: IDMSTUNE - 5 indexes/sets processed
DB002994 COM333: IDMSTUNE - processing completed
AutoCommit will COMMIT transaction
Command Facility ended with no errors or warnings

The following example directs the TUNE INDEX utility to adopt orphaned index records, rebalance and resequence the index. It also shows how to temporarily override the DMCL or subschema values for PAGE RESERVE and INDEX BLOCK CONTAINS.

TUNE INDEX FOR DBNAME EMPDEMO SUBSCHEMA EMPSS01
SET (EMP-NAME-NDX)
DEFAULT TUNE OPTIONS
   REBALANCE YES
   RESEQUENCE YES
   TEMPORARY INDEX UTILIZATION IS 80 %
   TEMPORARY PAGE RESERVE IS 15 PERCENT
   NOTIFY INTERVAL 1000;

The following report sample is produced by the TUNE INDEX utility.

TUNE INDEX FOR DBNAME EMPDEMO SUBSCHEMA EMPSS01
SET (EMP-NAME-NDX)
DEFAULT TUNE OPTIONS
   REBALANCE YES
   RESEQUENCE YES
   TEMPORARY INDEX UTILIZATION IS 80 %
   TEMPORARY PAGE RESERVE IS 15 PERCENT
   NOTIFY INTERVAL 1000;
Status = 0  SQLSTATE = 00000  Messages follow:
DB002994 COM333: IDMSTUNE - processing started
DB002994 COM333: IDMSTUNE - Indexes selected for processing:
DB002994 COM333: IDMSTUNE - EMP-NAME-NDX (IBC=32) in area EMPDEMO.EMP-DEMO-
The UNLOAD utility unloads records from one or more areas of the database in preparation for reloading. The UNLOAD utility does not reload data.

Note: You cannot use UNLOAD to unload an SQL database that contains keys longer than 256 bytes. Instead, use the REORG utility instead of UNLOAD to reorganize a large key database.

This article describes the following information:

- Authorization (see page 279)
- Syntax (see page 279)
- Parameters (see page 280)
- Usage (see page 281)
- JCL Considerations (see page 287)
- Example (see page 287)
- Sample Output (see page 287)

### Authorization

<table>
<thead>
<tr>
<th>To</th>
<th>You Need This Privilege</th>
<th>On</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unload an area</td>
<td>DBAREAD</td>
<td>The area</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Any area(s) with set connections to the area</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Any area(s) containing an index defined on records in the area</td>
</tr>
<tr>
<td>Unload a segment</td>
<td>DBAREAD</td>
<td>All areas of the segment</td>
</tr>
</tbody>
</table>

### Syntax

```
UNLOAD SEGMENT unload-segment-name
```
Parameters

- **unload-segment-name**
  Specifies the segment of the database to unload.

- **unload-subschema-name**
  Specifies the subschema that describes the database.
  By default, if an `unload-subschema-name` is not specified, it indicates that an SQL-type segment is to be unloaded. In this case, the UNLOAD utility builds the required subschema for unload processing.

- **area-name**
  Specifies an area in the unload segment to unload. The area must also exist in the reload segment. By default, if no areas are specified, all areas in the unload segment that are also defined in the unload subschema will be unloaded.

- **reload-segment-name**
  Specifies the segment in which the data will be reloaded.
  This parameter is passed in a control record at the beginning of the RELDCTL file to be used by the RELOAD utility.
  If an SQL-defined database is being unloaded, `segment-name` must be the same as the one specified in the SEGMENT clause.

- **reload-subschema-name**
  Specifies the subschema that describes the segment in which the data will be reloaded.
  Do not specify a reload subschema if unloading an SQL-type segment.
  This parameter is passed in a control record at the beginning of the RELDCTL file to be used by the RELOAD utility.
  By default, if a reload subschema is not specified, the subschema that was used to unload the data is used to reload the data.

- **reload-dmcl-name**
  Specifies the DMCL that defines the segment into which the data will be reloaded. See "The Unload/Reload DMCL" in the "Usage" section for additional information.
  This parameter is passed in a control record at the beginning of the RELDCTL file to be used by the RELOAD utility.
  By default, if a reload-dmcl-name is not specified, the DMCL used during the unload process is used. Unload uses the DMCL that is specified in the SYSIDMS parameter file. If no DMCL is specified in the SYSIDMS parameter file, the name IDMSDMCL is used.
Usage

How to submit the UNLOAD statement

You submit the UNLOAD utility only through the batch command facility. You must run the batch command facility in local mode.

Using UNLOAD on an SQL-defined database

You can use the UNLOAD utility statement to unload and reload an SQL-defined database to make these changes:

- Change area page ranges
- Change page size
  See `EXPAND PAGE (see page 64)` for complete information about changing a page size.
- Change the maximum number of records per page

The page changes are specified in the DMCL definition.

⚠️ **Note:** The modified DMCL must be available during the unload operation and during the reload operation.

You cannot make changes to table definitions in an SQL-defined database between the unload and reload steps.

Using UNLOAD on a non-SQL-defined database

You unload and reload a non-SQL-defined database in order to modify the database based on changes made to the schema or segment definition. These changes must be reflected in the subschema or DMCL used for the reload operation. An SQL-defined database must be reloaded into the same segment. You cannot reload it into another segment.

⚠️ **Note:** The modified DMCL or subschema must be available during the unload operation and during the reload operation.

You can make the following changes by using UNLOAD and RELOAD:

- Change area page ranges
- Change page size
  See `EXPAND PAGE (see page 64)` for complete information on changing a page size.
- Change page ranges for record types
- Reassign records to different areas
- Change record placements within an area
- Change record location modes
- Store VIA records by means of a different set (as long as they already participate in the set)
- Change compression and INDEX BLOCK CONTAINS options for indexes
- Change area and page-range assignments for system-owned indexes
- Change the maximum number of records per page

**UNLOAD and ASF databases**

If the UNLOAD utility is to be run against an ASF data or definition area, see the *CA IDMS ASF User Section* for more information.

**When not to use UNLOAD**

The areas processed by the UNLOAD utility cannot contain any logically deleted records. Therefore, do not use it until you have removed all logically deleted records with the CLEANUP utility statement.

The UNLOAD utility cannot be used against native VSAM files.

During UNLOAD and RELOAD processing, record formats are preserved, that is, the layout of data within a record and record positions within sets. This limits the modifications that you can make during an unload/reload operation. For example, you cannot use an unload/reload operation to:

- Remove a set
- Remove a record type
- Insert new data fields into a record
- Change the order of a sorted set
- Connect records to new sets
- Change or delete record IDs
- Change the size, location, or data type of sort or index keys
- Change a set from unordered to ordered

These type of changes require other utilities, such as RESTRUCTURE, and possibly user-written programs.

**Using UNLOAD and Mixed Page Groups**
UNLOAD cannot process mixed page groups. The subschemas specified for the UNLOAD cannot contain areas that reside in page groups other than the page group for the segment being unloaded. If the environment has an EXIT34 installed that exit will be invoked. If the CA supplied EXIT34 is used (RHDCUX34) and has not been altered a message will be produced indicating that an unqualified FIND/OBTAIN DBKEY command has been issued. If the exit has been modified to abort the associated run-unit when this type of command is encountered, the UNLOAD utility will be abnormally terminated. You must use multiple invocations of the utility to process areas in different page groups.

**UNLOAD and the DCMT VARY SEGMENT/AREA PERMANENT**

If the UNLOAD utility is to be run with the purpose of changing an area's page range and that change includes changing the area's low page, it is recommended that none of the DCMT VARY SEGMENT/AREA commands using the PERMANENT option be issued against the original area(s). The PERMANENT feature is implemented by carrying the area's low-page number in the journals across cycles of the CV. Changing an area's low-page will prohibit future cycles of the CV to properly identify the area once the new page range is implemented.

If a DCMT VARY SEGMENT/AREA PERMANENT command is still in effect when the new page range is implemented, the area's usage-mode at startup will be determined by the value specified in the DMCL. The entry in the journals for the old area's page range will remain until the next format of the journals.

**UNLOAD and the CLEANUP utility**

If the area being unloaded contains logically deleted records, run the CLEANUP utility before running unload.

⚠️ **Note:** For more information about making these kinds of changes, see the *CA IDMS Database Administration Section*.

**How UNLOAD Works**

During UNLOAD processing, an area sweep is performed, unloading all CALC, DIRECT, and unowned VIA records. At the same time, the area sweep extracts information for all indexes regardless of whether or not the index has been specified for the data area being unloaded.

After unloading the owner of a VIA set, the area sweep is interrupted to unload all the members of the set.

When unloading a CALC record for which duplicates are allowed, the CALC set is walked in the prior direction to determine the relative position of the record with respect to other record occurrences with duplicate keys.

**Non-SQL-defined sets can be partially unloaded**

Records in an area being unloaded or reloaded can have set connections (as owners or members) to records in areas not being processed in the same run.
For example, a CALC record in an area being unloaded can own a VIA member in an area that is not being unloaded.

The UNLOAD utility will keep track of the set connections, and the connections will be rebuilt when the records are reloaded.

**Note:** UNLOAD cannot process mixed page groups and will issue an error message if mixed page groups are encountered. You must use multiple invocations of the utility to process different page groups.

**SQL-defined segments**

Area and table stamps are unloaded during the UNLOAD steps of an SQL-defined database and are reloaded during RELOAD processing. For this reason, you must format the affected areas using the FILE option between the unload and reload operations.

The one exception to this is when unloading and reloading the DDLCATLOD area in a segment defined for SQL. In this case, either format by area or use the INSTALL STAMPS utility before reloading the data.

**DIRECT records and their VIA clusters**

Since DIRECT records have been placed on a specific page by a user, it may be difficult for the UNLOAD utility to determine where they should be placed in the new area. If a DIRECT record's old page exists in the new area, the RELOAD attempts to place the occurrence on the same page. However, if the old page does not exist in the new page range, the occurrence is targeted to the low page of the new area.

If a DIRECT record owns a VIA record, the members are stored in their new area proportional to their owner's position in its old area. If the owners and members reside in the same area and the page range has been extended or completely changed, the VIA cluster may not be on the same page as the owner. If this is not acceptable, the user may want to consider using a user-written program to unload the database and the FASTLOAD utility to reload the area.

**The Unload/Reload Subschemas**

The UNLOAD utility uses information in the dictionary when processing an SQL-defined database. When processing a non-SQL-defined database, the unload and reload subschemas must include the items listed next.

**The unload subschema**

The subschema used in an unload operation must:

- Include the areas being unloaded
- Include all areas containing records with set connections to records in the areas being unloaded
- Define all record types in the areas being unloaded
Define all sets in which the record types participate
Define all record types which can participate in the defined sets

The reload subschema

The subschema used in a reload operation (specified by RELOAD INTO..USING) must:
- Define the same record types and sets as the unload subschema
- Include the areas to be reloaded and all areas containing records with set connections to those areas.
- Allow all included areas to be readied in exclusive update mode

Unloading a dictionary

When unloading a dictionary, the IDMSNWKU subschema should be used for all areas, except DDLCAT, DDLCATX, and DDLCATLOD.

The DDLCAT and DDLCATX areas should be unloaded and reloaded using the IDMSCATZ subschema.

The DDLCATLOD area should be unloaded and reloaded using the IDMSCATL subschema. Before reloading the DDLCATLOD area in a segment defined for SQL, you must install stamps either by formatting by area or using the INSTALL STAMPS utility.

The Unload/Reload SEGMENT

The SEGMENT used in the unload and reload operation must include:
- The area(s) being unloaded.
- All areas with physical connections to the area(s) being unloaded. This includes:
  - Areas containing indexes on records or tables being unloaded.
  - Areas with set connections to areas being unloaded.
  - Areas containing tables related through a constraint to tables being unloaded.

The Unload/Reload DMCL

The DMCL used in the unload and reload operations must include the segment whose areas are being unloaded.

The UNLOAD utility must establish access to DMCLs for the unloading of the original database and for determining the page ranges of the database areas to be used by the subsequent RELOAD utility. The DMCL(s) to be used by each step of the process is determined as follows.
The unload operation will use the DMCL specified in the SYSIDMS file for the UNLOAD utility jobstep. If no DMCL is specified in the SYSIDMS file, the DMCL named IDMSDMCL will be used.

**Reloading the new database**

The DMCL specified in the RELOAD clause of the UNLOAD utility parameter will be used. If a DMCL is not specified, the DMCL to be used by the UNLOAD operation will be used to determine record placement in the new DMCL and is passed on to the RELOAD utility.

If the area definitions between the original database and the reload database are different, the unload and reload DMCLs must have different names, or the area definition must reside in different segments within the same DMCL.

**General Procedure for UNLOAD and RELOAD**

Take the following steps to unload and reload a database:

1. Create the new subschema or DMCL reflecting the changes to be made.
2. Back up all areas to be unloaded and also areas linked to those areas through sets or linked constraints.
3. Execute the CLEANUP utility if required.
4. Execute the UNLOAD utility.
5. Format the areas into which the data will be reloaded.
6. Execute the RELOAD utility.
7. Back up the same areas as in step 2.

This procedure may vary slightly depending on the type of change being made.

**Output files**

The following output files and sort parameters are generated by the UNLOAD utility for use by the RELOAD utility statement:

<table>
<thead>
<tr>
<th>File Contents</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS A record occurrence descriptor for each unloaded record occurrence.</td>
<td>For each occurrence descriptor: 24 bytes plus the size of the data portion of the record occurrence.</td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>A set descriptor for each unloaded record occurrence and for each record occurrence that is not unloaded but has a set connection to an unloaded record occurrence.</td>
<td>For each set descriptor: 24 bytes plus 12 times the number of pointers associated with the record, up to 32 pointers. If the record has more than 32 pointers, it will get an additional set descriptor.</td>
</tr>
<tr>
<td>SYS An index key occurrence descriptor for each index on a record occurrence.</td>
<td>For each index on each record occurrence: 32 bytes plus the size of the symbolic keys in the index.</td>
</tr>
</tbody>
</table>
CA IDMS Reference - 19.0

SYS Sort parameters
PC
H

RE   A control record with information about 60 bytes
LD   the subschema, DMCL, and segment to be
CTL  used by RELOAD.

A set descriptor for each set in the subschema.  60 bytes

JCL Considerations

When you submit an UNLOAD utility through the batch command facility, the JCL to execute the facility must include statements to define the following:

- Files containing the areas to be processed
- Journal file(s), which can be dummied
- Output files

⚠️ Note: For more information about the generic JCL used to execute the batch command facility, see the section for your operating system in this section.

Example

The following example directs the UNLOAD utility to unload an area in an SQL-defined segment named USERDB.

```
unload segment userdb
area "emp-area"
reload into userdb dmcl newdmcl;
```

Sample Output

The UNLOAD utility generates the following listing after successful completion of the previous example.

```
IDMSBCF mm/dd/yy PAGE 1  IDMS Batch Command Facility

SET BATCH WIDTH PAGE 80 ;
Status = 0
UNLOAD SEGMENT USERDB
AREA "EMP AREA"
RELOAD INTO USERDB DMCL NEWDMCL ;
UT001002 UNLOAD WILL USE SS DMCL IDMSDMCL DBNAME USERDB
```
UNLOCK

The UNLOCK utility removes the external lock on an area.

This article describes the following information:

- Authorization (see page 289)
- Syntax (see page 289)
- Parameters (see page 289)
- Usage (see page 289)
- JCL Considerations (see page 289)
- Examples (see page 290)
- Sample Output (see page 290)
Authorization

<table>
<thead>
<tr>
<th>To</th>
<th>You Need This Privilege</th>
<th>On</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unlock an area</td>
<td>DBAWRITE</td>
<td>The area</td>
</tr>
</tbody>
</table>

Syntax

```
UNLOCK AREA segment-name.area-name SEGMENT segment-name
```

Parameters

- **area**
  
  Directs the UNLOCK utility to remove the lock, if present, on the specified area.

- **segment-name**
  
  Specifies the name of the segment associated with the area.

- **area-name**
  
  Specifies the name of the area.

- **SEGMENT segment-name**
  
  Specifies the name of the segment to be unlocked.

Usage

**How to submit the UNLOCK statement**

You submit the UNLOCK statement only through the batch command facility. When submitting UNLOCK statements, you must run the batch command facility in local mode.

JCL Considerations

When you submit an UNLOCK statement through the batch command facility, the JCL to execute the facility must include statements to define the file containing the first page of the area to be processed.

**Note:** For more information about the generic JCL used to execute the batch command facility, see the section for your operating system in this section.
Examples

Unlock by area

The following example directs the UNLOCK utility to remove the external lock on the EMPDEMO.EMP-DEMO-REGION area.

unlock area empdemo.emp-demo-region;

Unlock by segment

The following example directs the UNLOCK utility to remove the external lock on the EMPDEMO.EMP-DEMO-REGION segment.

unlock segment empdemo;

Sample Output

Unlock by area

When UNLOCK processing is completed, the following listing is generated.

```
IDMSBCF mm/dd/yy PAGE 1
IDMS Batch Command Facility

UNLOCK AREA EMPDEMO.EMP-DEMO-REGION;
Status = 1 Extended Reason Code = 2367 Messages follow:
DB002367 C1M353: Warning: area EMPDEMO.EMP-DEMO-REGION was not locked.

AutoCommit will COMMIT transaction
Command Facility ended with warnings
```

Unlock by segment

When UNLOCK processing is completed, the following listing is generated.

```
IDMSBCF nn.n mm/dd/yy PAGE 1
CA IDMS Batch Command Facility

UNLOCK SEGMENT EMPDEMO;
Area EMPDEMO.EMP-DEMO-REGION was not locked
Area EMPDEMO.INS-DEMO-REGION was not locked
Area EMPDEMO.ORG-DEMO-REGION was not locked
Status = 0 SQLSTATE = 00000
```

⚠️ Note: For more information about area locks, see the CA IDMS Database Administration Section.
UPDATE STATISTICS

The UPDATE STATISTICS utility updates statistical information maintained in the dictionary for one or more tables. CA IDMS/DB uses this information when determining the optimal access strategy for processing SQL statements.

This article describes the following information:

- Authorization (see page 291)
- Syntax (see page 291)
- Parameters (see page 291)
- Usage (see page 292)
- JCL Considerations (see page 294)
- Example (see page 294)
- Sample Output (see page 294)

Authorization

<table>
<thead>
<tr>
<th>To update statistics for</th>
<th>You need this privilege</th>
<th>For</th>
</tr>
</thead>
<tbody>
<tr>
<td>A table</td>
<td>ALTER</td>
<td>The table</td>
</tr>
<tr>
<td>All tables in an area</td>
<td>DBAREAD</td>
<td>The area</td>
</tr>
<tr>
<td>Non-SQL schema</td>
<td>ALTER</td>
<td>All tables processed in the schema</td>
</tr>
</tbody>
</table>

Syntax

```
UPDATE STATISTICS
   FOR TABLE table-identifier
       , schema-name.
       , segment-name.area-name
       , SCHEMA schema-name
       , AREA area-name
       , SAMPLE percent
```

Parameters

- **FOR**
  Identifies the tables or areas for which the UPDATE STATISTICS utility is to update statistics.
TABLE
Specifies one or more SQL-defined tables and non-SQL defined tables for which the UPDATE STATISTICS utility is to update statistics.

- `schema-name`
  Specifies the name of the schema associated with the named table.

- `table-identifier`
  Specifies the identifier of a base table defined in the dictionary.

AREA
Updates statistics for all the tables in one or more SQL-defined areas. If one of the specified areas is non-SQL-defined, error message DB002316 is displayed which indicates that the area is NOT a relational area. In this case, specify the areas through the SCHEMA-clause of the UPDATE STATISTICS utility.

- `segment-name`
  Specifies the name of the segment associated with the area.

- `area-name`
  Specifies the name of the area.

SAMPLE
Specifies the percentage of the pages in an area that the UPDATE STATISTICS utility is to examine when calculating statistical information about one or more tables in the area.

- `percent`
  Specifies an integer in the range 1 through 100.
  By default, if you do not specify a percentage, the UPDATE STATISTICS utility will examine all the pages in the specified area.

SCHEMA `schema-name`
Identifies the schema for which statistics are updated. The identified schema must be SQL-defined and can reference a non-SQL-defined schema.

AREA `area-name`
Identifies one or more areas of the identified schema for which statistics are to be updated.

Usage

How to submit the UPDATE STATISTICS statement
You submit the UPDATE STATISTICS statement to CA IDMS/DB either online or through the batch command facility. If you submit it through the batch command facility, you should execute it through the central version, because the dictionary is updated as part of processing the statement.

Journal in local mode
If you are running CA IDMS/DB in local mode, you can journal while updating statistics.

These statistics are updated
When updating statistics for a table, the UPDATE STATISTICS utility also updates statistics about:

- The indexes defined on the table
- The referential constraints in which the table is the referenced table
- The area associated with the table

**Which pages are examined**

When you specify a percentage of less than 100 in the SAMPLE parameter, the UPDATE STATISTICS utility selects the pages to be examined from across the entire area. For example, if you specify SAMPLE 50, the UPDATE STATISTICS utility examines every other page throughout the entire area, rather than every page in the first half of the area.

**UPDATE STATISTICS extrapolates the statistics**

When calculating statistics based on less than 100 percent of the pages in an area, the UPDATE STATISTICS utility extrapolates the values for the entire area from the values based on the percentage examined. For example, if you specify SAMPLE 50, and the 50 percent of the pages that are examined in the area contain 80 rows of the table, the UPDATE STATISTICS utility assumes the table has 160 rows.

The statistics stored in the dictionary reflect the extrapolated values, not the raw data.

**UPDATE STATISTICS for IDMSNTWK**

When updating statistics for an SQL-defined schema that references the non-SQL-defined schema IDMSNTWK, the UPDATE STATISTICS utility will only process area DDLDML. If an area other than DDLDML is specified using the AREA clause, a warning will be issued indicating there are no tables to process.

**UPDATE STATISTICS for native VSAM files**

When updating statistics for an SQL-defined schema that references a non-SQL-defined schema containing native VSAM files, the job abends with error-message DB002300 and DBIO Error code 3077. The UPDATE STATISTICS module IDMSCOLS collects statistics of some information found on a BDAM DB page (for example, total space available count) that cannot be found on a native VSAM page. In this case, the DBIO Error code 3077 indicates that an attempt was made to run an invalid CA IDMS utility against a native VSAM file.

**UPDATE STATISTICS for a non-SQL-defined schema**

When updating statistics for an SQL-defined schema that references a non-SQL-defined schema, statistics are stored in the dictionary (DDLDML) where the non-SQL schema is defined. For a system-owned index key that does not contain group elements, statistics are kept for each index column. If the index key contains a group element, a DB003202 warning message is issued and only the statistics of the first index column are updated.
JCL Considerations

When you submit an UPDATE STATISTICS statement to CA IDMS/DB through the batch command facility, the JCL to execute the facility must include either:

- A SYSCTL file to direct execution to the central version
- or

- Statements to define:
  - The areas containing the table(s) being examined
  - The dictionary containing the table definitions
  - The journal files of the DMCL you are using

Example

Updating statistics for specified tables

The following UPDATE STATISTICS statement updates statistics for the MONTHLY_BUDGET and PROPOSED_BUDGET tables in the PROD schema. Since no percentage has been specified, the UPDATE STATISTICS utility will examine all the pages in the area associated with each table when calculating the statistics.

```sql
update statistics for table prod.monthly_budget, prod.proposed_budget;
```

Updating statistics for all the tables in an area

The following UPDATE STATISTICS statement updates statistics for all the tables in the SQLDEMO.EMPLAREA area.

```sql
update statistics for area sqldemo.emplarea;
```

Sample Output

After successful completion of the UPDATE STATISTICS statement on the SQLDEMO.EMPLAREA area shown previously, the following listing is produced.

```
IDMSBCF mm/dd/yy PAGE 1 IDMS Batch Command Facility

UPDATE STATISTICS FOR AREA SQLDEMO.EMPLAREA;
Status = 0
AutoCommit will COMMIT transaction
Command Facility ended with no errors or warnings
```
VALIDATE

The VALIDATE utility checks linked and unlinked referential constraints for a referencing table, making sure that referenced tables exist and contain the appropriate column values.

<table>
<thead>
<tr>
<th>Type of VALIDATE</th>
<th>What it does</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete VALIDATE</td>
<td>Runs both STEP1 and STEP2</td>
</tr>
<tr>
<td>Stepped VALIDATE</td>
<td>Runs one step at a time with intermediate file sorting required between STEP1 and STEP2</td>
</tr>
</tbody>
</table>

This article describes the following information:

- Authorization (see page 295)
- Syntax (see page 295)
- Parameters (see page 296)
- Usage (see page 297)
- JCL Considerations (see page 298)
- Example (see page 299)
- Sample Output (see page 299)

Authorization

<table>
<thead>
<tr>
<th>To</th>
<th>You need this privilege</th>
<th>On</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate a table</td>
<td>SELECT</td>
<td>The table</td>
</tr>
<tr>
<td>Validate the tables in an area</td>
<td>SELECT</td>
<td>All tables in the area</td>
</tr>
<tr>
<td>Validate the tables in a segment</td>
<td>SELECT</td>
<td>All tables in the segment</td>
</tr>
</tbody>
</table>

Syntax

Syntax for complete VALIDATE

```
VALIDATE TABLE schema-name.table-identifier AREA segment-name.area-name SEGMENT segment-name NOTIFY notify-record-count ERRORS error-count CONTINUE
```
Syntax for stepped VALIDATE

```
VALIDATE
  ┌───────────────────ˌ,─┐
  ┌─┬─────────┬──┬──┐
  │STEP1  ─┘  │  │  │
  │ AREA segment-name.area-name  │  │  │
  └─┴──────────┘  │  │  │
  │STEP2  ──────────────────────┘porter
  ┌─┬────────┬─┐
  │NOTIFY notify-record-count  │
  └───┘
  ┌─┬──┐
  │ERRORS  │error-count  │
  └───┘
  └─┬─────┐
      │CONTINUE  │
```

**Note:** Only one LOAD, BUILD, or VALIDATE statement can be performed during one execution of the Batch Command Facility (IDMSBCF).

**Parameters**

- **TABLE**
  Identifies the referencing table to validate.

- **schema-name**
  Specifies the name of the schema that defines the table.

- **table-identifier**
  Specifies the identifier of the table.

- **AREA**
  Identifies the area containing tables to be validated. All referencing tables in the area will be validated.

- **segment-name**
  Specifies the name of the segment containing the area.

- **area-name**
  Specifies the name of the area.

- **SEGMENT**
  Identifies the segment containing tables to be validated. All referencing tables in the segment will be validated.

- **segment-name**
  Specifies the name of the segment.

- **NOTIFY**
  Directs the VALIDATE utility to send a message to the operator whenever a specified number of rows are processed.
  The message states the phase and step currently being executed and the number of records processed.
notify-record-count
Specifies the number of rows to validate before sending a message.

ERRORS
Directs the VALIDATE utility to either continue the validation when errors are detected or stop after a specified number of errors are detected.
By default, processing will not stop when errors are detected.
Detected errors are listed in the report generated by the VALIDATE utility.

error-count
Specifies the number of errors to detect before terminating.

CONTINUE
Directs the VALIDATE utility to continue processing regardless of the number of errors detected. CONTINUE is the default.

STEP1
Validates only linked referential constraints and unlinked index-to-index referential constraints.
If other unlinked referential constraints are detected, VALIDATE STEP1 produces an intermediate work file to be used as input to VALIDATE STEP2. If no such file is produced, you do not need to run VALIDATE STEP2.
If you do not specify a STEP number, the VALIDATE utility will validate all linked and unlinked referential constraints, and is considered a complete VALIDATE.

STEP2
Validates all unlinked referential constraints except index to index referential constraints.

Usage

How to submit the VALIDATE statement
You submit the VALIDATE statement only through the batch command facility. You must run the batch command facility in local mode.

When to use VALIDATE
If you have loaded a group of tables using a phased or stepped LOAD and built the indexes and relationships of the tables specifying NO VALIDATE in the BUILD statement, use VALIDATE to ensure that referencing tables have valid references.

You can also use the VALIDATE utility at any time to validate the referential constraints of a table.

VALIDATE utility uses intermediate work files
STEP1 of the VALIDATE utility produces an intermediate work file to be used by STEP2. If you run a complete VALIDATE without separating STEP1 from STEP2, data is sorted in the intermediate file between the steps automatically. If you run a stepped VALIDATE, you must run the intermediate sorts.
Note: When running a complete VALIDATE, SYS002 and SYS003 must point to the same intermediate file. When running a stepped VALIDATE, SYS002 and SYS003 must point to different intermediate files. The data that is output in SYS003 by STEP1 is input to STEP2 in SYS002.

When not to use VALIDATE

If you loaded the tables with a complete LOAD, or if you did not specify NO VALIDATE in the BUILD statement, then the validation has already been done. There is no need to run the VALIDATE utility.

If the tables have no referential constraints, there is no need to run the VALIDATE utility.

Note: For more information about referential constraints, see the CA IDMS Database Administration Section.

JCL Considerations

When you submit a VALIDATE statement through the batch command facility, the JCL to execute the facility must include statements to define the following:

- Dictionary containing table definitions
- Files containing the areas associated with the referencing tables to be processed
- Intermediate work files
- Sort work files are needed if doing a complete VALIDATE

Sorting intermediate work files

If you run the validate process in steps or phases, use the sort parameters in the SYSPCH file to sort the intermediate files.

The following table shows the output of the steps of the VALIDATE process:

<table>
<thead>
<tr>
<th>Step</th>
<th>Output</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEP 1</td>
<td>SYS003</td>
<td>For each record: (MAX SORT CONTROL KEY SIZE) + (MAX FOREIGN KEY SIZE) + 24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYSPCH contains sort parameters</td>
</tr>
<tr>
<td>STEP 2</td>
<td>SYS003</td>
<td>For each record: (MAX SORT CONTROL KEY SIZE) + (MAX FOREIGN KEY SIZE) + 24</td>
</tr>
</tbody>
</table>
Step | Output
--- | ---
SYSPCH contains sort parameters

Note: For more information about the generic JCL used to execute the batch command facility, see the section specific to your operating system.

Example

The following example instructs the VALIDATE utility to perform a validation check against sample tables M and M2. The validation was not performed when the BUILD utility was run against them.

```
validate table load.m,
    load.m2
errors continue;
```

Sample Output

The following listing was generated after validating sample tables M and M2 in the previous example.

```
IDMSBCF

*DEBUG IDMS OFF
CONNECT TO SYSCAT;
Status = 0
SET BATCH
    HEADINGS OFF WIDTH PAGE 79 UNDERLINE '-'
    SQLCODE ERROR
    COMPRESS ON;
UNLOCK AREA SYSSQL.DDLCAT;
Status = 1    Extended Reason Code = 2367    Messages follow:
DB002367 C1M353: Area SYSSQL.DDLCAT was not locked.
UNLOCK AREA SYSSQL.DDLCATX;
Status = 1    Extended Reason Code = 2367    Messages follow:
DB002367 C1M353: Area SYSSQL.DDLCATX was not locked.

-- **** Load data into Tables ****
*DEBUG IDMS ON
VALIDATE TABLE LOAD.M,
    LOAD.M2
ERRORS CONTINUE;
IDMSLOAD - volser    VALIDATE INDEXES STEP 1       yy-mm-dd-hh.mm.ss
IDMSLOAD - 0 records processed for table LOAD.M
IDMSLOAD - 0 records processed for table LOAD.M2
IDMSLOAD - 3 intermediate records were written to SYS003
IDMSLOAD -     largest SYS003 record size is 56 characters
IDMSLOAD -     VALIDATE INDEXES STEP 1    processing completed
AutoCommit will COMMIT transaction
Command Facility ended with warnings
```
Utility Programs (CA IDMS/DB Utilities)

This topic describes the following utility programs:
- IDMSCALC Utility (see page 300)
- IDMSDBAN (see page 301)
- IDMSDIRL (see page 316)
- IDMSLOOK (see page 320)
- IDMSRPTS (see page 330)
- IDMSRSTC (see page 349)

IDMSCALC Utility

The IDMSCALC utility is a subroutine that can be called from a user-written subroutine to determine the target page of a record, based on a user-supplied CALC key.

It is typically used to optimize the loading of data by allowing you to presort input in target page sequence.

- Usage (see page 300)
- Calling the IDMSCALC Routine (see page 300)

Usage

How IDMSCALC works

IDMSCALC is implemented as a called subroutine. The utility returns to a user-written program a target page number for storage of a CALC record, based on a page range and CALC key value supplied by the program. The user program, which can be written in any language supporting a call statement, must build a single five-field fullword-aligned argument as outlined in the following table, then call IDMSCALC, passing the argument. IDMSCALC must be link edited with the calling program.

Calling the IDMSCALC Routine

The following example shows how to call the IDMSCALC routine from a user-written program.

```
01 CALC-PARMS.
  05 CALC-PAGE-TARGET PIC S9(9) COMP.
  05 CALC-PAGE-RANGE-HIGH PIC S9(9) COMP.
  05 CALC-PAGE-RANGE-LOW PIC S9(9) COMP.
  05 CALC-KEY-LENGTH PIC S9(4) COMP.
  05 CALC-KEY PIC X(16).

MOVE 75001 TO CALC-PAGE-RANGE-LOW.
MOVE 75101 TO CALC-PAGE-RANGE-HIGH.
MOVE 16 TO CALC-KEY-LENGTH.
```
MOVE ‘SMITH’ TO CALC-KEY.
CALL ‘IDMSCALC’ USING CALC-PARMS.
DISPLAY ‘TARGET PAGE IS ’ CALC-PAGE-TARGET.

The IDMSCALC argument

The following table outlines the five-field argument that a calling program must pass to IDMSCALC.

<table>
<thead>
<tr>
<th>Field</th>
<th>Usage</th>
<th>Size</th>
<th>COBOL Picture</th>
<th>Description of Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(Output) Binary 4 bytes PIC 9(9) COMP</td>
<td>Target page number for storage of the record.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(Input) Binary 4 bytes PIC 9(9) COMP</td>
<td>Number of the highest page on which the record can be stored.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(Input) Binary 4 bytes PIC 9(9) COMP</td>
<td>Number of the lowest page on which the record can be stored.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>(Input) Binary 2 bytes PIC 9(4) COMP</td>
<td>Length, in bytes, of the CALC key value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>(Input) Character 1-256 bytes PIC X(nnn)</td>
<td>Value of the CALC key.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

⚠️ **Note:** The information in fields 2 and 3 of IDMSCALC must match the database definition for the record type as specified in the schema.

Input

Input to the IDMSCALC utility consists of the IDMSCALC argument with fields 2-5 initialized by the calling program.

Output

The IDMSCALC utility returns a target page number for storage of a CALC record.

IDMSDBAN

The IDMSDBAN database analysis utility analyzes the characteristics and structure of a CA IDMS/DB database (both non-SQL and SQL-defined). The utility provides information useful for system tuning, database structuring, and capacity planning.

IDMSDBAN also verifies the integrity of:

- Page structures
- Line indexes
Record lengths
Record locations
Set connections for:
  - Chained sets
  - Constraints
  - Indexed sets
  - The CALC set
  - Variable-length-record fragment chains

This article describes the following information:

- Authorization (see page 302)
- Syntax (see page 302)
- Input Parameter Statements (see page 303)
- Usage (see page 307)
- JCL Considerations (see page 311)
- Example (see page 312)
- Sample Output (see page 312)

### Authorization

<table>
<thead>
<tr>
<th>To</th>
<th>You Need This Privilege</th>
<th>On</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analyze an area</td>
<td>DBAREAD</td>
<td>The area</td>
</tr>
<tr>
<td>Analyze a set, constraint, or table</td>
<td>DBAREAD</td>
<td>The area where the set, constraint, or table resides</td>
</tr>
</tbody>
</table>

### Syntax

**PROCESS statement**

Code on one line only.

```plaintext
PROCCESS SUBSCHEMA subschema-name subschema-specification SQL
```

```
LOCKed
UNLocked
```

*Expansion of subschema-specification*

```plaintext
DBNAME db-name OWNCHECK INComplete
```
REPORT statement

Code on one line only.

```plaintext
REPORT report-number TERSE
```

AREA statement

Code on one line only.

```plaintext
AREAs ALL area-specification
```

Expansion of area-specification

```plaintext
area-name segment-name.
```

PAGE start-page TO end-page

SET statement

Code on one line only.

```plaintext
SETS CONSTRAINT INDEX
```

```plaintext
ALL NOCALC NOVLR
```

```plaintext
set-name ON schema.table-id CALC
```

```plaintext
VLR
```

Input Parameter Statements

IDMSDBAN processing is controlled by the following input parameter statements. If coded, the three types of statements must be coded in the order shown. If not coded, all reports are provided for all areas and sets for the named subschema or SQL-defined database.

Statement descriptions

<table>
<thead>
<tr>
<th>Statement</th>
<th>Required/ Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROCESS</td>
<td>Required</td>
<td>For non SQL-defined database: Identifies subschema and DMCL module Specifies whether areas are to be locked before processing For SQL-defined database: Identifies that an SQL-defined database will be processed Specifies whether areas are to be locked before processing</td>
</tr>
<tr>
<td>REPORT</td>
<td>Optional</td>
<td>Controls generation of reports 2, 3, 4, and 5</td>
</tr>
<tr>
<td>AREA</td>
<td>Optional</td>
<td>Specifies area(s) to be processed</td>
</tr>
<tr>
<td>SET</td>
<td>Optional</td>
<td>Specifies set(s) or constraints and indexes to be processed</td>
</tr>
</tbody>
</table>

PROCESS statement

---

16-Jan-2018

303/493
The PROCESS statement identifies the database to be processed and specifies whether areas are to be locked before processing begins. One PROCESS statement is required for each IDMSDBAN run.

**Syntax**

Code on one line only.

```
<table>
<thead>
<tr>
<th>PROCESS</th>
<th>SUBSCHEMA subschema-name subschema-specification</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LOCKed</td>
</tr>
<tr>
<td></td>
<td>UNLocked</td>
</tr>
</tbody>
</table>

**Expansion of subschema-specification**

```
<table>
<thead>
<tr>
<th>DBNAME db-name</th>
<th>OWNCHECK</th>
<th>INComplete</th>
</tr>
</thead>
</table>
```

**Parameters**

- **SUBSCHEMA subschema-name**
  Identifies the subschema containing the areas, sets, and records to be processed. The subschema must contain complete descriptions for all of its sets and records.

  **Note:** When processing the DDLCAT and DDLCATX areas, you must use the SUBSCHEMA parameter and the IDMSCATZ subschema name.

  When processing the DDLCATLOD area you must use the SUBSCHEMA parameter and the IDMSCATL subschema name.

- **DBNAME db-name:**
  Identifies the name of the database to bind to at run time. If no DBNAME is specified, the default is the DBNAME specified in the SYSIDMS parameter file.

- **OWNCHECK**
  Indicates that checks for ownerless loops in chained sets is to be performed. If this parameter is omitted, IDMSDBAN will not check for chained sets that form a loop but do not include an owner record occurrence.

  **Note:** This check will increase the run time of the utility and may require that the allocation for the SYS002 file in the IDMSDBN2 step be increased.

- **INComplete**
  Specifies that the named subschema does not include all record types. INCOMPLETE suppresses messages that would otherwise appear for record types not included in the subschema.

- **SQL**
  Specifies that an SQL-defined database is to be processed. Based upon the segments identified on the AREA statement and the sets or constraints identified on the SET statement, IDMSDBAN will build a subschema to process an SQL-defined database.
When processing the DDLCAT and DDLCATX areas, you must use the SUBSCHEMA parameter and the IDMSCATZ subschema name.

When processing the DDLCATLOD area you must use the SUBSCHEMA parameter and the IDMSCATL subschema name.

- **LOCKed/UNLocked**
  Specifies whether IDMSDBAN is to lock the areas to be processed:
  - **LOCKED** (default) directs IDMSDBAN to lock the areas during IDMSDBN1 processing. When you specify LOCKED, the areas involved cannot be updated by another application during IDMSDBAN execution.
  - **UNLOCKED** directs IDMSDBAN not to lock the areas to be processed. When UNLOCKED is specified, the areas involved can be updated during IDMSDBAN execution.

**REPORT statement**

The optional REPORT statement controls the generation of reports 2, 3, 4, and 5. IDMSDBAN always generates reports 1 and 1A.

By default, if you do not supply a REPORT statement or if you supply a REPORT statement without any parameters, IDMSDBAN will generate all reports.

**Syntax**

Code on one line only.

```plaintext
REPORT report-number [TERSE]
```

**Parameters**

- **report-number:**
  Specifies the number of the report to be generated. Valid report numbers are 1, 1A, 2, 3, 4, and 5. If no REPORTS statement is coded or if a REPORTS statement is provided without any report numbers, all reports are generated.

- **TERSE**
  Applies to reports 3, 4, and 5 only, as follows:
  - TERSE causes report 3 to be produced only for sets that have at least one non-empty set occurrence.
  - TERSE causes report 4 to be produced only for record types with at least one occurrence in the database.
  - TERSE causes report 5 to be produced only for sets with at least one occurrence in the database.

**AREA statement**
The AREA statement specifies the areas to be processed. If no AREA statement is coded, all areas in the named subschema or segment are processed.

Syntax

Code on one line only.

```
AREAs     ALL      area-specification
```

Expansion of area-specification

```
segment-name area-name PAGE start-page TO end-page
```

Parameters

- **AREAs**
  
  Specifies the area(s) to process.
  
- **ALL**
  
  Specifies that all areas included in the specified subschema are to be processed.
  
  ALL is the default.
  
  You cannot use the ALL option if SQL was selected on the PROCESS statement.

- **segment.**
  
  For SQL-defined databases, identifies the name of the segment to be processed. You must specify `segment-name` if SQL was selected on the PROCESS statement.
  
  When processing a non-SQL-defined database, do not specify `segment-name`.
  
  `Segment-name` must be a 1 through 8-character value.

- **area-name:**
  
  Identifies the name of an area to be processed.
  
  `Area-name` must be a 1 through 18-character value.

- **PAGE start-page-n TO end-page-n:**
  
  Specifies the range of pages to be processed within the named area. The specified page range must be included within the area page range.
  
  By default, if you do not specify a page range, the entire area will be processed.
  
  If you do specify a page range, be sure that the page range selected includes all database pages that can contain owner or member records of all sets to be processed. Otherwise, they won't be processed.
  
  `Start-page` and `end-page` must contain numeric values and can be from 1 through 10-characters in length.

SET Statement

The SET statement specifies the sets, constraints, or indexes to be processed.

Syntax

Code on one line only.

```
SETS CONSTRAINT INDEX ALL NOCALC NOVLR set-name ON schema.table-id
```
Parameters

- **SETS/CONSTRAINT/INDEX**
  Identifies the sets, constraints, or indexes to process.
  The keywords SET, CONSTRAINT, and INDEX are synonyms and can be used interchangeably.
  All owners and members must be included in the area to be analyzed and page ranges must be identified in the AREA statement or no sets or constraints are processed.

- **ALL**
  Indicates that all of the following sets associated with a specified subschema or segment are to be processed for either a non-SQL or SQL-defined database:
  - All chained and indexed sets defined in the subschema specified in the PROCESS statement
  - All constraints within the specified segment
  - The CALC set
  - Sets of variable-length record fragments located within the areas specified in the AREA statements
  ALL is the default.

- **NOCALC**
  Removes the CALC set from the list of sets specified by ALL.

- **NOVLR**
  Removes the set of variable length record chains from the list of sets specified by ALL.

- **set-name:**
  Identifies the name of a set to be processed.

- **ON schema.table-id:**
  If SQL was specified on the PROCESS statement, you must identify the name of the SQL schema and table identifier the constraint or index is associated with. You do not specify a schema.table-id for non-SQL-defined databases. *Schema* is a 1 through 18-character value and *table-id* is a 1 through 18-character value.

- **CALC**
  Directs IDMSDBAN to process the CALC set within the specified area(s).

- **VLR**
  Directs IDMSDBAN to process all sets of variable-length record fragment chains located within the specified area(s).
Input to the IDMSDBAN utility consists of statements to control the utility processing.

Output

The IDMSDBAN utility generates message listings.

Execution mode

You can execute the IDMSDBAN utility in local mode only.

Usage considerations

Specifying dictionary name for SQL-defined databases

When processing an SQL-defined database, you must specify the dictionary name or catalog segment that IDMSDBAN connects to using the SYSIDMS DBNAME parameter (not the DICTNAME parameter).

**Note:** For more information about the SYSIDMS parameter file, see the *CA IDMS Common Facilities Section*.

Subschema

The subschema that describes the database areas to be processed must include all member record types for all sets included in the subschema.

Processing DDLCAT, DDLCATX, and DDLCATLOD areas

When processing the DDLCAT and DDLCATX areas, you must use the PROCESS statement SUBSCHEMA parameter and the IDMSCATZ subschema name.

When processing the DDLCATLOD area you must use the PROCESS statement SUBSCHEMA parameter and the IDMSCATL subschema name.

Areas

By default, IDMSDBAN locks the areas to be processed. If IDMSDBAN attempts to lock an area that is already locked, the utility terminates. To prevent this, perform one of the following actions:

- Either vary offline or vary to retrieval the areas to be processed before running IDMSDBAN.
- Specify UNLOCKED in the PROCESS parameter statement.

**Note:** If you specify UNLOCKED and the database is updated by another application during IDMSDBAN processing, the statistics in reports 2, 3, 4, and 5 may be inaccurate, and the messages in reports 1 and 1A may be misleading.

Page ranges
The page ranges selected for processing must include all database pages that can contain owner or member records of any set(s), constraints, or indexes to be processed.

If you specify a set, constraint, or index with an owner or member record outside the specified pages, IDMSDBAN will terminate with an error.

**Indexed sets**

If an indexed set is specified for processing, processing must also be requested for all other indexed sets for which the SR8 records can occur on the same pages as the SR8 records for the specified set.

**IDMSDBAN has two parts**

IDMSBN1 and IDMSBN2.

**Intermediate work file size**

The size of the intermediate work files will vary depending on the size and complexity of the database being analyzed.

- IDMSBN1 will generate one output record for each pointer position in each record for each set being processed. The file can be written to tape or disk; however, to ensure adequate space, tape is recommended.

- IDMSBN2 uses an intermediate work file to hold records it is working on. For every record it receives from IDMSBN1, it will produce up to two intermediate work records. If the OWNCHECK option has been specified, it is necessary for the utility to produce these records for each occurrence of a record participating in a chained set.

**Tape management systems and IDMSDBAN work files**

The SYS002 work file used by IDMSBN2 is written to and read from repeatedly. If this file is placed on a tape, then your tape management system may prevent IDMSBN2 from overwriting the file after it has written to the file the first time. If the tape management system installation defaults allow, specify a zero retention period for the work file and/or specify DISP=(NEW,DELETE) for the work file.

**If problems are detected**

If the messages issued by IDMSBN indicate that problems exist in the database, the database should not be updated in between the time of the IDMSBN run and the time that the problems are corrected.

**IDMSBN1**

IDMSBN1 sweeps each specified area. For each area, IDMSBN1:

- Collects statistics
  - Detailed statistics for each area and record type.
  - Summary statistics for user-defined chained and indexed sets.
Generates an intermediate work file
IDMSDBN1 generates input to IDMSDBN2.
Each record in the file represents a pointer in a set connection between two records. For example, one record might represent the next pointer between two member records in a user-defined set; another might represent the connection between the root of a variable-length record and the first fragment of the record.
Each record includes the database keys of the two connected records. The database key of the record that contains the pointer to the other record is the FROM database key. The database key of the record to which the first record points is the TO database key.

Output

<table>
<thead>
<tr>
<th>Report</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1: Messages</td>
<td>Required</td>
<td>Lists input parameters used in the run.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lists all messages issued by IDMSDBN1. The messages:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Report errors in the parameter input</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Trace the processing of areas</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Define any unexpected conditions detected by the program</td>
</tr>
<tr>
<td>#2: Area Information</td>
<td>Optional</td>
<td>Provides detailed area statistics. The report includes histograms of space availability and of data records per page.</td>
</tr>
<tr>
<td>#3: Set Statistics</td>
<td>Optional</td>
<td>Presents summary set statistics.</td>
</tr>
<tr>
<td>#4: Record Information</td>
<td>Optional</td>
<td>Provides detailed statistics for each record type in each area being analyzed. The report includes a histogram of data records per page.</td>
</tr>
</tbody>
</table>

**IDMSDBN2**

As a performance enhancement for the IDMSDBN2 step, the SIZE=E99999999 parameter has been added to the various sorts that are executed. However, some sort packages do not allow the usage of the SIZE= parameter on their SORT statements. If your sort package does allow the specification of the SIZE= parameter, you can reduce the run time of IDMSDBN2 by coding SORTSIZE=ON in the jobstep's SYSIDMS input file.

IDMSDBN2 does the following:

- Verifies set integrity. This is done in three steps.
  1. Reading the intermediate work file from IDMSDBN1
  2. Iteratively massaging and sorting the records
  3. Creating chains
     A *chain* is a path that originates at a record located at a FROM database key and terminates at a record located at a TO database key.
IDMSDBN2 concatenates chains until a closed loop is created. A loop is created when the record at the last TO database key matches the record at the first FROM database key. Each chain created by IDMSDBN2 is associated with a set. IDMSDBN2 verifies the integrity of sets by ensuring that each chain is complete and contains one and only one owner.

- Collects detailed set statistics.

### Output

<table>
<thead>
<tr>
<th>Report</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1A: Messages</td>
<td>Required</td>
<td>Lists all messages issued by IDMSDBN2. The messages: Trace the processing of the chain file. Report inconsistencies detected during chain processing.</td>
</tr>
</tbody>
</table>

| #5: Set Analysis Information | Optional | Provides detailed statistics for each set type processed. Chained sets -- The report includes histograms of chain length, of page changes, and of pages used to store the set. Sets of variable-length-record fragments -- The report includes a histogram of pages used to store the set. Indexed sets owned by a user record -- The report includes histograms of SR8 usage in set occurrences, of members in set occurrences, and of SR8 levels in set occurrences. (SR8 records are internal index records.) |

### JCL Considerations

The JCL to execute the IDMSDBAN utility program must include statements to define:

- For IDMSDBN1:
  - For SQL databases, you must specify the name of the catalog to be processed on the SYSIDMS DBNAME parameter and not the DICTNAME parameter.
  - The files that map to the areas to be processed.
  - The SYSIPT file containing input parameters.
  - SYS002 contains output for use by IDMSDBN2 (this file is known to IDMSDBN2 as SYS001).

- For IDMSDBN2:
  - SYS001 containing the output from IDMSDBN1 (this file was known to IDMSDBN1 as SYS002).
  - SYS002 is a temporary storage file.
  - SORTWKnn are sort work files. The number and size depends on the sort package you use.
  - SORTMSG containing sort output messages.
SYSLST is the destination for the reports the utility creates. If this DD points to a sequential file, pre-allocate the file before the IDMSDBN2 step and ensure the file has a disposition within the step of DISP=MOD. If you fail to do so, Report 1A will not be present on the file at step completion.

⚠️ **Note:** For more information about the generic JCL used to execute the batch command facility, see the section for your operating system in this section.

### Example

**No REPORT parameter specified**

If you run IDMSDBAN with the following input parameters, all areas and all sets in the EMPSS01 subschema will be processed and all reports will be produced. All reports are produced because a REPORT parameter is not specified.

```
process subschema empss01 dbname empdemo unlocked;
```

**With REPORT parameters**

If you run IDMSDBAN with the following input parameters, only the named areas and sets will be processed as follows:

- The entire EMP-DEMO-REGION area
- Pages in the range 5007102 through 5007149 in the ORG-DEMO-REGION
- Reports 1, 1A, 3 and 5 will be generated
- Report 3 will be produced for all of the named sets
- Report 5 will be produced only for the named sets that have at least one occurrence in the database

```
process subschema empss01 incomplete
  report 3
  report 5 terse
  area emp-demo-region
  area org-demo-region page 5007102 to 5007149
  set dept-employee
  set office-employee
  set emp-name-ndx;
```

### Sample Output

The following reports are generated when no REPORT parameters are specified as in the first example.

**Report 1: Messages Phase I**
### Report 3: Set Statistics

**IDMSDBAN - DATA BASE ANALYSIS**

**REPORT 3: SET STATISTICS**

<table>
<thead>
<tr>
<th>DATE</th>
<th>TIME</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>00/00/2023</td>
<td>00:00:00</td>
<td>1</td>
</tr>
</tbody>
</table>

**SET:** EMP-EMPOSITION  
**MODE:** CHAIN  
**ORDER:** FIRST  
**SUBSCHEMA:** EMPSS01

<table>
<thead>
<tr>
<th>NAME</th>
<th>SET MEMBERSHIP</th>
<th>NUMBER OF RECORDS</th>
<th>EMPTY SETS</th>
<th>UNCONNECTED MEMBERS</th>
<th>LOCATION MODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER: EMPLOYEE</td>
<td></td>
<td>56</td>
<td>0</td>
<td>CALC</td>
<td></td>
</tr>
<tr>
<td>MEMBERS: EMPOSITION</td>
<td>MA</td>
<td>68</td>
<td>0</td>
<td>VIA EMP-EMPOSITION</td>
<td></td>
</tr>
</tbody>
</table>

**AVERAGE MEMBERS PER SET FOR NON-EMPTY SETS:**

### Report 4: Record Data

**IDMSDBAN - DATA BASE ANALYSIS**

**REPORT 4: RECORD DATA**

<table>
<thead>
<tr>
<th>DATE</th>
<th>TIME</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>00/00/2023</td>
<td>00:00:00</td>
<td>1</td>
</tr>
</tbody>
</table>

**RECORD:** EMPLOYEE  
**RECORD ID:** 415  
**LOCATION MODE:** CALC  
**SUBSCHEMA:** EMPSS01  
**AREA:** EMPDEMO.EMP-DEMO-REGION  
**FLR/VR:** FLR

| DATA LENGTH | 116 |
| RECORD LENGTH (DATA AND PREFIX): | 188 |
| PAGE RANGE FOR RECORD IN AREA - START PAGE: | 75,003 |
| END PAGE: | 75,100 |
| PAGES ALLOCATED FOR RECORD IN AREA: | 98 |
| PAGE RANGE ANALYZED - START PAGE: | 75,003 |
| END PAGE: | 75,100 |
| PAGES PROCESSED FOR RECORD IN AREA IN THIS RUN: | 98 |
| NUMBER OF SMP WITHIN PAGES PROCESSED: | 0 |
| PAGE SIZE IN BYTES: | 4,276 |
| BYTES USED FOR THIS RECORD TYPE IN PAGES PROCESSED: | 10,976 |
| BYTES AVAILABLE IN PAGES PROCESSED: | 392,736 |
| NUMBER OF RECORDS OF THIS TYPE IN PAGES PROCESSED: | 56 |
Report 1A: Messages Phase II

<table>
<thead>
<tr>
<th>IDMSDBAN PHASE II</th>
<th>REPORT 1A: MESSAGES</th>
<th>DATE</th>
<th>TIME</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>/yy</td>
<td>20240900</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>599001</td>
<td>PHASE II PROCESSING BEGIN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>599003</td>
<td>END PASS</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>599001</td>
<td>FROM-RECORDS WRITTEN TO SORT</td>
<td>630</td>
<td></td>
<td></td>
</tr>
<tr>
<td>599002</td>
<td>TO-RECORDS WRITTEN TO SORT</td>
<td>630</td>
<td></td>
<td></td>
</tr>
<tr>
<td>599006</td>
<td>INDEX SR8 DESCRIPTORS WRITTEN</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>599007</td>
<td>INDEX DOWN DESCRIPTORS WRITTEN</td>
<td>58</td>
<td></td>
<td></td>
</tr>
<tr>
<td>599008</td>
<td>INDEX UP DESCRIPTORS WRITTEN</td>
<td>58</td>
<td></td>
<td></td>
</tr>
<tr>
<td>599009</td>
<td>INDEX NO-UP MEM DESCRIPTORS WRITTEN</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>599010</td>
<td>INDEX NO-UP DOWN DESCRIPTORS WRITTEN</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>599003</td>
<td>END PASS</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>599001</td>
<td>FROM-RECORDS WRITTEN TO SORT</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>599002</td>
<td>TO-RECORDS WRITTEN TO SORT</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>599005</td>
<td>INDEX PATHS WRITTEN TO SORT</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>599006</td>
<td>INDEX SR8 DESCRIPTORS WRITTEN</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>599007</td>
<td>INDEX DOWN DESCRIPTORS WRITTEN</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>599008</td>
<td>INDEX UP DESCRIPTORS WRITTEN</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>599003</td>
<td>END PASS</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>599001</td>
<td>FROM-RECORDS WRITTEN TO SORT</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>599002</td>
<td>TO-RECORDS WRITTEN TO SORT</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>599005</td>
<td>INDEX PATHS WRITTEN TO SORT</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>599006</td>
<td>INDEX SR8 DESCRIPTORS WRITTEN</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>599007</td>
<td>INDEX DOWN DESCRIPTORS WRITTEN</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>599008</td>
<td>INDEX UP DESCRIPTORS WRITTEN</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Report 5: Set Analysis

<table>
<thead>
<tr>
<th>IDMSDBAN PHASE II</th>
<th>REPORT 5: SET ANALYSIS DATA</th>
<th>DATE</th>
<th>TIME</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>/yy</td>
<td>20240900</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SET: EMP-EMPOSITION  MODE: CHAIN  ORDER: FIRST  SUBSCHEMA: EMPSS01
OWNER: EMPLOYEE
MEMBERS: EMPOSITION

CHAIN LENGTH HISTOGRAM

<table>
<thead>
<tr>
<th>CHAIN LENGTH</th>
<th>NUMBER OF SETS</th>
<th>PERCENT OF TOTAL SETS</th>
<th>GRAPH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
More Information

- For more information about database structures analyzed by IDMSDBAN, see the CA IDMS Database Administration Section.

- For more information about database tuning, see the CA IDMS Database Design Section.

IDMSDIRL

The dictionary load utility, IDMSDIRL, loads into a dictionary the components required to describe the dictionary itself and the components that describe the security information stored in the dictionary.

The CA-supplied internal schema components include:

- Element and record definitions required by the schema components

- IDMSNTWK schema, which describes the dictionary itself and two associated subschemas; IDMSNWKA and IDMSNWKG

- IDMSSECS and IDMSSECU schemas, which describe the security information stored in the dictionary and the IDMSSECS and IDMSSECU subschemas used for security processing

- Additionally, IDMSDIRL can optionally remove these components from a dictionary without loading new definitions

Specifically, IDMSDIRL performs the following functions when loading the components:

- Establishes the default quotation character for the dictionary

- Adds to the dictionary the element and record definitions used by the schema components

- Adds to the dictionary the definitions for the IDMSNTWK schema and its associated subschemas: IDMSNWKA and IDMSNWKG

- Adds to the dictionary the definitions for the IDMSSECS schema and its associated subschema: IDMSSECS

- Adds to the dictionary the definitions for the IDMSSECU schema and its associated subschema: IDMSSECU
Optionally connects the S-010 record occurrences just created for schemas IDMSNTWK, IDMSSECS, and IDMSSECU to the OOAK-S set so that data dictionary reports include these definitions.

Note: If the dictionary to be loaded is empty (formatted), you must first input the dictionary into DDDL which will populate the necessary CA-internal definitions into the dictionary.

This article describes the following information:

- Syntax (see page 317)
- Input Parameter Statements (see page 317)
- Usage (see page 318)
- JCL Considerations (see page 319)
- Examples (see page 320)
- Sample Output (see page 320)

### Syntax

**DBL-QUOTE statement**

Code on one line only.

```plaintext
 DBL-QUOTE
```

**SCHEMA-CONNECT statement**

```plaintext
 SCHEMA-CONNECT
```

**SCHEMA-DELETE statement**

```plaintext
 SCHEMA-DELETE
```

### Input Parameter Statements

Parameter statement descriptions

IDMSDIRL utility processing can be controlled by optional input parameter statements:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBL-QUOTE</td>
<td>Used to override the default quotation character for the data dictionary.</td>
</tr>
<tr>
<td>SCHEMA-CONNECT</td>
<td>Used to connect the CA-supplied internal schemas to the OOAK-S set.</td>
</tr>
<tr>
<td>Statement</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SCHEMA-DELETE</td>
<td>Used to delete any CA-supplied internal schemas in the dictionary. Does not perform a dictionary load.</td>
</tr>
</tbody>
</table>

**DBL-QUOTE statement**

The DBL-QUOTE statement makes the double quotation mark (""") the default quotation character for the data dictionary.

By default, if you do not specify DBL-QUOTE, the default quotation character for the dictionary is the single quotation mark ("').

**Syntax**

Code on one line only.

```
DBL-QUOTE
```

**SCHEMA-CONNECT statement**

The SCHEMA-CONNECT statement connects the CA-supplied internal schemas to the OOAK-S set.

By default, if you do not specify SCHEMA-CONNECT, the IDMSNTWK, IDMSSECS, and IDMSSECU schemas are not connected to the OOAK-S set.

**Syntax**

Code on one line only.

```
SCHEMA-CONNECT
```

**SCHEMA-DELETE statement**

The SCHEMA-DELETE statement deletes any CA-supplied internal schemas (IDMSNTWK, IDMSSECS, and IDMSSECU) and associated subschemas from the dictionary.

Any orphaned records and elements are deleted.

The SCHEMA-DELETE statement does not result in a dictionary load.

**Syntax**

```
SCHEMA-DELETE
```

**Usage**

**Input**

Input to the IDMSDIRL utility consists of:
The file containing the definitions to be loaded. This file is supplied on the CA IDMS/DB installation media.

Optional statements to control the utility processing. Each of these statements must be coded on a separate line beginning in column 1. There is no end-of-statement terminator for these statements.

Output

The IDMSDIRL utility generates a message listing.

Batch operating mode

You can execute the IDMSDIRL utility either in local mode or under the central version. Local mode is recommended.

When to use IDMSDIRL

IDMSDIRL is typically used to load one data dictionary at a site.

The data dictionary is used by the following CA products:

- CA OLQ, when reporting on the dictionary
- CA Culprit, when producing one of the following reports:
  - CA ADS reports (AReports)
  - System generation reports (CReports)
  - IDD reports (DReports)

Performance consideration for SCHEMA-CONNECT

When reporting on the dictionary, the IDMSRPTS utility walks the OOAK-S set, if you specify SCHEMA=ALL. If the CA-supplied internal schemas are connected to the OOAK-S set, the reports will include all components of these schemas. Excluding the CA-supplied internal schemas from the OOAK-S set allows IDMSRPTS to bypass these components.

JCL Considerations

The JCL to execute the IDMSDIRL utility program must include statements to define:

- The file containing the input definitions
- The file containing the dictionary into which the definitions are to be loaded
Examples

The following example directs the IDMSDIRL utility to add the definitions for the IDMSSECU schema and its associated subschema IDMSSECU to the dictionary and to connect the CA-supplied internal schemas to the OOAK-S set. Additionally, by default, the single quotation mark is established as the quotation character.

```
schema-connect
```

Sample Output

The IDMSDIRL utility program generates the following listing after processing the previous input.

```
IDMSDIRL - DATA DIRECTORY LOAD UTILITY RELEASE nn.n volser
OOAK ALREADY EXISTS
END OF IDMS DIRECTORY LOAD - IDMSDIRL
```

IDMSLOOK

The IDMSLOOK load module print utility reports on the contents of selected load modules. Using IDMSLOOK, you can report on:

- The contents of a database name table module
- Area, record, and set information in a subschema load module
- Area, record, set, file, and physical characteristics for a subschema bound to a specific database
- File, area, journal, and buffer information in a DMCL load module together with its associated database name table information
- The names of the RCM modules characteristics of the tables referenced by the access module
- A hexadecimal dump of a specified load module in a format suitable for display at a 3270-type terminal
- The date/time stamps of the component modules in a specified load module
- The external value of internal date/time stamps
- The contents of the product intent module RHDCPINT

This article describes the following information:

- Syntax (see page 321)
- Parameters (see page 322)
- Usage (see page 325)
- JCL Considerations (see page 325)
- Examples (see page 325)
- Sample Output (see page 326)

## Syntax

### Expansion of `look-command`

```plaintext
look-command
```

#### Expansion of `am` command

- `AM= access-module-name`  
  - `AM PROGRAM= access-module-name`
  - `BIND SQL SEGMENT=segment-name,DBNAME=database-name`  
    - `BIND SUBSCHEMA= subschema-name`  
      - `DATABASE=database-name`
  - `DATES= load-module-name`
  - `DATETIME`
  - `DATETIME STAMP= hex-internal-datetime-stamp`
  - `DBTABLE`
  - `DBTABLE= dbtable-name`
  - `DMCL`
    - `ALL`
    - `SORTED`
    - `PAGE`
  - `EXTERNAL DATETIME= external-datetime-value`
  - `FIND=hex-string`
  - `HELP`
  - `LOAD=OPSYS`
  - `OPTION Flags`
  - `PROGRAM= load-module-name`
  - `PROGRAM POOL`
  - `RCM PROGRAM= RCM-module-name`
  - `RHDCPINT`
  - `STORAGE POOL`
  - `SUBSCHEMA= subschema-name`  
    - `fib-opt`

### Expansion of `fib-opt`

- `ALL`
- `RADIX`
- `NOFILES`
- `BOUNDFILES`
Parameters

- **ALL**
  Specifies that some commands display additional information about the DMCL or subschema entities reported. Currently supported by DMCL and FIB related reports.

- **AM=access-module-name**
  Displays the contents of the RCMs included in an ACCESS MODULE and the IB50 built into the ACCESS MODULE.

- **AM PROGRAM**=
  Displays a core dump of an ACCESS module.

- **BIND SQL SEGMENT**=
  Displays the logical and physical attributes for areas, tables, constraints, and indexes for a segment of an SQL-defined database. The output is similar to that of the BIND SUBSCHEMA function.
  
  - **segment-name**
    Specifies the name of the segment that contains the SQL database areas.

- **DBNAME=database-name**
  Specifies the name of the database that contains the segment where the catalog for the SQL definitions reside.

- **BIND SUBSCHEMA**=
  Displays the logical and physical attributes of the subschema.
  
  - **subschema-name**
    Specifies the name of a subschema load module.

- **DBNAME**=
  Required unless you are binding to an originally built release 10.x subschema that is being converted to Release 12.0 format.

  - **database-name**
    Specifies the name of a database.

- **BOUNDFILES**
  Specifies that only files connected to bound areas are displayed. This option is available for FIB-related reports.

- **DATES**=
  Displays the DATE/TIME stamps of the components of a specified load module.

  - **load-module-name**
    Specifies the name of a load module.

- **DATETIME**
  Displays the current Date/Time.

- **DATETIME STAMP**=
  Displays the external value of an internal date/time stamp.
- **hex-internal-datetime-stamp**
  The 16 hexadecimal digits that make up the internal representation of the date/time stamp.

- **DBTABLE**
  Displays the contents of the default DBNAME table that is associated with the DMCL.

- **DBTABLE=**
  Displays the contents of the named DBNAME table.

- **dbtable-name**
  Specifies the name of a DB table load module

- **DMCL**
  Reports the contents of the current DMCL module.

- **ALL**
  Optionally, produces the following information in addition to the standard information provided on the DMCL report:
  - The date each area definition was last updated
  - A history of the last date and time that an area was affected by a DCMT VARY DMCL command

- **SORTED**
  Sorts DMCL information by area name.

- **SORTED PAGES**
  Sorts DMCL information by page range.

- **RADIX**
  Specifies that each area displays the number of bits reserved for a dbkey line number in hex. This replaces the Data Sharing flag value.

- **EXTERNAL DATETIME=external-datetime-value**
  The 26 characters that make up the external representation of the date/time stamp. The format is yyyy-mm-dd-hh.mm.ssfffffff.

  - yyyy specifies the year. yyyy must be an integer in the range 0001 through 9999.
  - mm specifies the month within the year. mm must be an integer in the range 01 through 12.
  - dd specifies the day within the month. dd must be an integer in the range 01 through 31.
  - hh specifies the hour on a 24-hour clock. hh must be an integer in the range 00 through 23.
  - mm specifies the number of minutes past the hour. mm must be an integer in the range 00 through 59.
  - ss specifies the number of seconds past the minute. ss must be an integer in the range 00 through 59.
  - ffffffff specifies the number of millionths of a second past the specified second.
- **FIND**=*hex-string*
  Displays the program name and offset into the program where the address was found. Hex-string is the 8 hexadecimal digits of the address to be searched for. The address must reside in one of the programs that reside in the PROGRAM POOL.

- **HELP**
  Displays the parameters supported by the LOOK task.

- **LOAD**=*OPSYS*
  Specifies that load modules are loaded from an operating system load library rather than from a dictionary load area. This statement affects the loading technique used by SUBSCHEMA, DATES, PROGRAM, and RCM PROGRAM statements, and when coded, must precede any of these statements.

- **NOFILES**
  Suppresses the display of DMCL files, buffers, journals, and DBTABLE data. This option is available for FIB-related reports.

- **OPTION FLAGS**
  Displays all the optional APARs that have been activated in the current RHDCOPTF module.

- **PROGRAM**=*load-module-name*
  Displays the DATE/TIME stamp of all the components that make up the load module. Also provides a core dump of the load module.

- **PROGRAM POOL**
  Displays the contents of the PROGRAM POOL. Shows the program name, entry point address, load address, use count, and size of the program.

- **RADIX**
  Specifies that commands that display DMCL area information display the number of bits in a dbkey in hex that is reserved for a dbkey. This replaces the data sharing flag status.

- **RCM PROGRAM**=*load-module-name*
  Displays a core dump of an RCM MODULE.

- **RHDCPINT**
  Displays a list of all products you can license along with the product intent status if a RHDCPINT module is found in the library concatenation.

- **STORAGE POOL**
  Displays the contents of the STORAGE POOL. Shows the storage address, storage size, task number that acquired the storage, owner of the storage, and storage type.

- **SUBSCHEMA**=*subschema-name*
  Displays the logical attributes of the subschema.
Usage

Input

Input to the IDMSLOOK utility consists only of statements to control the utility processing.

Output

The IDMSLOOK utility generates a printout listing that includes the requested reports.

Batch operating mode

You can execute the IDMSLOOK utility in local mode only.

Coding considerations

You can include multiple input statements in a single run of IDMSLOOK. However, each input statement must be on a separate line.

None of the statements is required.

Online processing

Most of the functions of IDMSLOOK can be executed online through the DC task code LOOK.

Note: For more information on using DC tasks, see the CA IDMS System Tasks and Operator Commands Section.

JCL Considerations

For more information about the JCL used to execute IDMSLOOK, see the section for your operating system in this section.

Examples

Requesting subschema information

The following SUBSCHEMA statement directs IDMSLOOK to return a report on the subschema load module EMPSS01. The report will include information on the logical and physical attributes of the EMPSS01 subschema when it is bound to database name EMPDEMO.

bind subschema=empss01,dbname=empdemo

Requesting DMCL information
The following statement directs IDMSLOOK to report all information on a DMCL (page range, size, file mappings, and so on, as well as date and time history) and to sort the information by area name.

dmcl all sorted

**Requesting a hexadecimal dump**

The following PROGRAM statement directs IDMSLOOK to return a hexadecimal dump of the load module RTPRG001, along with the date/time stamps of the object modules included in RTPRG001.

```plaintext
program=rtpgr001
```

### Sample Output

IDMSLOOK starts a new page in the message listing for each input parameter statement processed in the run. At the top of the page, IDMSLOOK prints the parameter statement. Then IDMSLOOK prints the report requested by the statement.

#### Report requested by subschema

```
IDMSLOOK - Selection Parameter Follows:
SUBSCHEMA=EMPSS01

    EMPSS01 was LOADED from ---> APPLDICT
Entry Point Offset +0 : Reentrant : AMODE 31 : RMODE ANY

SUBSCHEMA=EMPSS01
Compiled=yyyy-mm-dd 15.24.58
Subschema Structure is Network and Unbound

<table>
<thead>
<tr>
<th>Area Name</th>
<th>Segment</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMP-DEMO-REGION</td>
<td>n/a</td>
</tr>
<tr>
<td>INS-DEMO-REGION</td>
<td>n/a</td>
</tr>
<tr>
<td>ORG-DEMO-REGION</td>
<td>n/a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Record Name</th>
<th>Stored</th>
<th>Rec ID</th>
<th>Area Name</th>
<th>Data Length</th>
<th>Prefix Length</th>
<th>Procedures</th>
</tr>
</thead>
<tbody>
<tr>
<td>COVERAGE</td>
<td>VIA</td>
<td>400</td>
<td>INS-DEMO-REGION</td>
<td>20</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>DENTAL-CLAIM</td>
<td>VIA</td>
<td>405</td>
<td>INS-DEMO-REGION</td>
<td>936</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>DEPARTMENT</td>
<td>CALC</td>
<td>410</td>
<td>ORG-DEMO-REGION</td>
<td>56</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>EMPLOYEE</td>
<td>CALC</td>
<td>415</td>
<td>EMP-DEMO-REGION</td>
<td>129</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td>EMPOSITION</td>
<td>VIA</td>
<td>420</td>
<td>EMP-DEMO-REGION</td>
<td>32</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>EXPERTISE</td>
<td>VIA</td>
<td>425</td>
<td>EMP-DEMO-REGION</td>
<td>12</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>HOSPITAL-CLAIM</td>
<td>VIA</td>
<td>430</td>
<td>INS-DEMO-REGION</td>
<td>360</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>INSURANCE-PLAN</td>
<td>CALC</td>
<td>435</td>
<td>INS-DEMO-REGION</td>
<td>132</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>JOB</td>
<td>CALC</td>
<td>440</td>
<td>ORG-DEMO-REGION</td>
<td>360</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>NON-HOSP-CLAIM</td>
<td>VIA</td>
<td>445</td>
<td>INS-DEMO-REGION</td>
<td>1,056</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>OFFICE</td>
<td>CALC</td>
<td>450</td>
<td>ORG-DEMO-REGION</td>
<td>76</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>SKILL</td>
<td>CALC</td>
<td>455</td>
<td>ORG-DEMO-REGION</td>
<td>76</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>SR1</td>
<td>VIA</td>
<td>1</td>
<td>n/a</td>
<td>4</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>SR6</td>
<td>VIA</td>
<td>6</td>
<td>n/a</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SR7</td>
<td>CALC</td>
<td>7</td>
<td>n/a</td>
<td>16</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>Chain Sorted-&gt;</td>
<td>CALC</td>
<td>7</td>
<td>Next,Prior</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Owner ----------&gt; SR1</td>
<td></td>
<td>Next=00 Prior=84</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Member ----------&gt; SR6</td>
<td></td>
<td>Next=00 Prior=84</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Member ----------&gt; SR7</td>
<td></td>
<td>Next=00 Prior=84</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Member ----------&gt; SKILL</td>
<td></td>
<td>Next=00 Prior=84</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Member ----------&gt; OFFICE</td>
<td></td>
<td>Next=00 Prior=84</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Member ----------&gt; JOB</td>
<td></td>
<td>Next=00 Prior=84</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

IDMSCOMP Before STORE
IDMSCOMP Before MODIFY
IDMSCOMP After GET

Ckey Offset=16 Length=16 Data Type=Character
Ckey Offset=20 Length=4 Data Type=Numeric (Unsigned)
Ckey Offset=16 Length=3 Data Type=Character
```
### CA IDMS Reference - 19.0

<table>
<thead>
<tr>
<th>Ckey</th>
<th>Offset</th>
<th>Length</th>
<th>Data Type</th>
<th>Member</th>
<th>Next</th>
<th>Prior</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>28</td>
<td>4</td>
<td>Numeric (Unsigned)</td>
<td>INSURANCE-PLAN</td>
<td>00</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>3</td>
<td>Character</td>
<td>EMPLOYEE</td>
<td>00</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td>72</td>
<td>4</td>
<td>Numeric (Unsigned)</td>
<td>DEPARTMENT</td>
<td>00</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>4</td>
<td>Numeric (Unsigned)</td>
<td>COVERAGE-CLAIMS</td>
<td>00</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>3</td>
<td>Character</td>
<td>EMPLOYEE</td>
<td>00</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td>72</td>
<td>4</td>
<td>Numeric (Unsigned)</td>
<td>DEPARTMENT</td>
<td>00</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>4</td>
<td>Numeric (Unsigned)</td>
<td>COVERAGE-CLAIMS</td>
<td>00</td>
<td>84</td>
</tr>
</tbody>
</table>

#### Report requested by DMCL

⚠️ **Note:** For data sharing installations, the "Shr" column in the generated report can have one of the following values:

- blank -- no data sharing enabled for area.
- D -- data sharing has been defined in the DMCL definition of the area, but it is not currently being shared.
- R -- data sharing is currently active for the identified area.
- The RADIX option replaces the "Shr" column with a radix value.

<table>
<thead>
<tr>
<th>IDMSLOOK - OPSYS=Z/OS</th>
<th>Release XX.X Service pack X</th>
<th>tape XXXXXX</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMCL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMCL=574DMCL</td>
<td>Runtime Size--&gt; 13008 (77,832 Bytes)</td>
<td></td>
</tr>
<tr>
<td>This DMCL uses dbtable R120DBTB</td>
<td>Compiled Size--&gt; 0BD7C (48,588 Bytes)</td>
<td></td>
</tr>
<tr>
<td>Date Last Critical Change=2008-04-30</td>
<td>Date Last Updated=2008-04-30</td>
<td></td>
</tr>
<tr>
<td>Data Created=2007-05-25</td>
<td>Date Last Updated=2008-04-30</td>
<td></td>
</tr>
<tr>
<td>Dynamic File Allocation – on</td>
<td>Dynamic File Allocation – on</td>
<td></td>
</tr>
<tr>
<td>Data Sharing is active – on connectivity no abend</td>
<td>Data Sharing is active – on connectivity no abend</td>
<td></td>
</tr>
<tr>
<td>Data Sharing Lock Entries - 4</td>
<td>Data Sharing Lock Entries - 4</td>
<td></td>
</tr>
<tr>
<td>Data Sharing Default Shared Cache - IDMSCACHE00002</td>
<td>Data Sharing Default Shared Cache - IDMSCACHE00002</td>
<td></td>
</tr>
<tr>
<td>Memory Cache Location ANYWHERE Storage Limit OPSYS</td>
<td>Memory Cache Location ANYWHERE Storage Limit OPSYS</td>
<td></td>
</tr>
</tbody>
</table>

16-Jan-2018 327/493
CA IDMS Reference - 19.0

### Network Area

<table>
<thead>
<tr>
<th>Area Name</th>
<th>Page</th>
<th>Low</th>
<th>High</th>
<th>Page Size</th>
<th>DONAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBCR.BRANCHTEL</td>
<td>15</td>
<td>680,001</td>
<td>685,012</td>
<td>4,000</td>
<td>BRANCHA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BRANCHB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BRANCHC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BRANCHD</td>
</tr>
<tr>
<td>NETWORK area</td>
<td>On STARTUP go Update</td>
<td>On WARMSTART use current status</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Definition date last critical change=2007-05-25 10.40.48</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Page Reserve size</td>
<td>0</td>
<td></td>
<td></td>
<td>1,984</td>
<td></td>
</tr>
<tr>
<td>Max Records Per Page</td>
<td></td>
<td></td>
<td></td>
<td>255</td>
<td></td>
</tr>
<tr>
<td>Page Range Symbolic is BRANCHTEL Value is 680,001---&gt;685,012</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| DBCR.ACCTHIST      | 15    | 690,001 | 740,040  | 2,932     | ACCOUNTA |
|                    |       |         |          |           | ACCOUNTB |
|                    |       |         |          |           | ACCOUNTC |
|                    |       |         |          |           | ACCOUNTD |
|                    |       |         |          |           | ACCOUNTE |
| NETWORK area       | On STARTUP go Update | On WARMSTART use current status |
| Definition date last critical change=2007-05-25 10.40.48 |
| Page Reserve size  | 0     |         |          | 1,450     |        |
| Max Records Per Page |      |         |          | 255       |        |
| Page Range Symbolic is ACCTHIST Value is 690,001--->740,040 |

| EMPDEMO.EMP-DEMO-REGION | 0    | 75,001  | 75,100   | 4,276     | EMPDEMO |
| NETWORK area            | On STARTUP go Update | On WARMSTART use current status |
| Definition date last critical change=2007-05-25 10.40.48 |
| Page Reserve size       | 0     |         |          | 2,122     |        |
| Max Records Per Page    |      |         |          | 255       |        |
| Page Range Symbolic is EMP-DEMO-REGION Value is 75,001--->75,100 |

| EMPDEMO.INS-DEMO-REGION | 0    | 75,101  | 75,150   | 4,276     | INSDEMO |
| NETWORK area            | On STARTUP go Update | On WARMSTART use current status |
| Definition date last critical change=2007-05-25 10.40.48 |
| Page Reserve size       | 0     |         |          | 2,122     |        |
| Max Records Per Page    |      |         |          | 255       |        |
| Page Range Symbolic is INS-DEMO-REGION Value is 75,101--->75,150 |

| EMPDEMO.ORG-DEMO-REGION | 0    | 75,151  | 75,200   | 4,276     | ORGDEMO |
| NETWORK area            | On STARTUP go Update | On WARMSTART use current status |
| Definition date last critical change=2007-05-25 10.40.48 |
| Page Reserve size       | 0     |         |          | 2,122     |        |
| Max Records Per Page    |      |         |          | 255       |        |
| Page Range Symbolic is ORG-DEMO-REGION Value is 75,151--->75,200 |

<table>
<thead>
<tr>
<th>Mem.- Shared File Name</th>
<th>DONAME</th>
<th>Type</th>
<th>Cache</th>
<th>Cache</th>
<th>Buffer Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBCR.ACCTA</td>
<td>ACCOUNTA</td>
<td>BDAM</td>
<td>No</td>
<td>Yes</td>
<td>DBCR_ACCT_BUFFER</td>
</tr>
<tr>
<td>DBCR.ACCTB</td>
<td>ACCOUNTB</td>
<td>BDAM</td>
<td>No</td>
<td>Yes</td>
<td>DBCR_ACCT_BUFFER</td>
</tr>
<tr>
<td>DBCR.ACCTC</td>
<td>ACCOUNTC</td>
<td>BDAM</td>
<td>No</td>
<td>Yes</td>
<td>DBCR_ACCT_BUFFER</td>
</tr>
<tr>
<td>DBCR.ACCTD</td>
<td>ACCOUNTD</td>
<td>BDAM</td>
<td>No</td>
<td>Yes</td>
<td>DBCR_ACCT_BUFFER</td>
</tr>
<tr>
<td>DBCR.ACOUNTE</td>
<td>ACCOUNTE</td>
<td>BDAM</td>
<td>No</td>
<td>Yes</td>
<td>DBCR_ACCT_BUFFER</td>
</tr>
<tr>
<td>DBCR.BRANCHA</td>
<td>BRANCHA</td>
<td>BDAM</td>
<td>No</td>
<td>Yes</td>
<td>DBCR_BRCH_BUFFER</td>
</tr>
<tr>
<td>DBCR.BRANCHB</td>
<td>BRANCHB</td>
<td>BDAM</td>
<td>No</td>
<td>Yes</td>
<td>DBCR_BRCH_BUFFER</td>
</tr>
<tr>
<td>DBCR.BRANCHC</td>
<td>BRANCHC</td>
<td>BDAM</td>
<td>No</td>
<td>Yes</td>
<td>DBCR_BRCH_BUFFER</td>
</tr>
<tr>
<td>DBCR.BRANCHD</td>
<td>BRANCHD</td>
<td>BDAM</td>
<td>No</td>
<td>Yes</td>
<td>DBCR_BRCH_BUFFER</td>
</tr>
<tr>
<td>DBCR.BRANCHTEL</td>
<td>BRANCHA</td>
<td>BDAM</td>
<td>No</td>
<td>Yes</td>
<td>DBCR_BRCH_BUFFER</td>
</tr>
<tr>
<td>DBCR.BRANCHB</td>
<td>BRANCHB</td>
<td>BDAM</td>
<td>No</td>
<td>Yes</td>
<td>DBCR_BRCH_BUFFER</td>
</tr>
<tr>
<td>DBCR.BRANCHC</td>
<td>BRANCHC</td>
<td>BDAM</td>
<td>No</td>
<td>Yes</td>
<td>DBCR_BRCH_BUFFER</td>
</tr>
<tr>
<td>DBCR.BRANCHD</td>
<td>BRANCHD</td>
<td>BDAM</td>
<td>No</td>
<td>Yes</td>
<td>DBCR_BRCH_BUFFER</td>
</tr>
</tbody>
</table>
CA IDMS Reference - 19.0

Display Component Module Information

The following LOOK command displays information about the component modules of program IDMSCHDC.

```
IDMSLOOK - 1 Selection Card Processed
```

```
DMCL Journals       Page Size   # of Pages
------------------- ------------------
SYSJRNL1            2,932       30,000
DMCL Buffers        Size     CV Buffers Type Size      Local Buffers Type Size
------------------- ------- ---------- ------- -------- ---------- ------- ---------- --------
DBCR_BRCH_BUFFER    4,000     5,000   OS      20,000,000 1,000   OS       4,000,000
DBCR_ACCT_BUFFER    2,932     5,000   OS     14,660,000 1,000   OS      14,660,000
LOG_BUFFER          9,076     30      OS    272,280      20     OS      181,520
DEFUALT_BUFFER      9,076     30      OS    272,280      20     OS      181,520
```

0 Bytes used for CV buffers in DC storage
94,951,040 Bytes used for CV buffers in OS storage
94,951,040 Bytes used for CV DMCL Buffers
18,760 Bytes used for LOCAL buffers in DC storage
19,113,520 Bytes used for LOCAL buffers in OS storage
19,132,280 Bytes used for LOCAL DMCL Buffers

```
DBNAME is *DEFAULT match on subschema is OPTIONAL
Subschema IDMSNWK? maps to IDMSNWK? using DBNAME ---> TSTDICT
Subschema EMPSS??? maps to EMPDEMO using DBNAME ---> EMPDEMO
Subschema EV?????? maps to ETOTSUS using DBNAME ---> TSTDICT
Subschema DBCR???? maps to DBCR???? using DBNAME ---> DBCR
```

```
Dbtable=R120DBTB   Compiled date=2007-05-16 17.18.23
The DEFAULT Dictionary is TSTDICT
```

```
IDMSLOOK - 1 Selection Card Processed
```
IDMSRPTS

The IDMSRPTS data dictionary reports utility provides information stored in a data dictionary. The utility is a useful tool for database administrators, application programmers, operations personnel, and managers. IDMSRPTS generates reports in four categories.

This article describes the following information:

- Report Types (see page 330)
- Syntax (see page 333)
- Statements (see page 334)
- Usage (see page 342)
- JCL Considerations (see page 343)
- Examples (see page 343)
- Sample Output (see page 344)

Report Types

This section describes the IDMSRPTS report types.

Non-database reports

Non-database reports provide information on occurrences of data dictionary entities that are not related to database processing.

You do not need to specify a schema, subschema, DMCL, database name table, or segment to get these reports.
<table>
<thead>
<tr>
<th>Full Report Name</th>
<th>Syntax Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Report Listing</td>
<td>GLBLRPT</td>
<td>Produces all non-database reports</td>
</tr>
<tr>
<td>Module/Process/Table Description Listings</td>
<td>MODLST</td>
<td>Produces all module, process, and table reports</td>
</tr>
<tr>
<td>Module Description Listing</td>
<td>MODULE</td>
<td>Reports on modules in the dictionary</td>
</tr>
<tr>
<td>ADS Process Description Listing</td>
<td>PROCESS</td>
<td>Reports on processes</td>
</tr>
<tr>
<td>Protocol Listing</td>
<td>PRTLST</td>
<td>Reports on protocols defined in the data dictionary</td>
</tr>
<tr>
<td>Record Copy Description Listing</td>
<td>RECCOPY</td>
<td>Reports on IDD-built records, reports, and transactions</td>
</tr>
<tr>
<td>OLQ Qfile Description Listing</td>
<td>QFILE</td>
<td>Reports on q-files defined in the data dictionary</td>
</tr>
<tr>
<td>Table Description Listing</td>
<td>TABLE</td>
<td>Reports on edit and code tables</td>
</tr>
<tr>
<td>User Listing</td>
<td>USER</td>
<td>Reports on users defined in the data dictionary</td>
</tr>
</tbody>
</table>

### Schema Reports

Schema reports provide information on specified schemas defined in the data dictionary.

<table>
<thead>
<tr>
<th>Full Report Name</th>
<th>Syntax Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema Report Listing</td>
<td>SCHRPT</td>
<td>Produces all schema reports</td>
</tr>
<tr>
<td>Area Listing</td>
<td>AREALST</td>
<td>Reports on areas defined in a specified schema</td>
</tr>
<tr>
<td>Program Cross-Reference Listing</td>
<td>PGMLST</td>
<td>Reports on subschemas compiled under a specified schema</td>
</tr>
<tr>
<td>Schema Record Description Listing</td>
<td>RECDES</td>
<td>Reports on record types defined in a specified schema</td>
</tr>
<tr>
<td>Schema Set Description Listing</td>
<td>SETDES</td>
<td>Reports on sets defined in a specified schema</td>
</tr>
</tbody>
</table>

### Subschema reports

Subschema reports provide information on specified subschemas defined in the data dictionary.

<table>
<thead>
<tr>
<th>Full Report Name</th>
<th>Syntax Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subschema Report Listing</td>
<td>SSCRPT</td>
<td>Produces all subschema reports</td>
</tr>
<tr>
<td>Subschema Data Directory Listing</td>
<td>DATDIR</td>
<td>Provides general information on record types copied into a specified subschema</td>
</tr>
<tr>
<td>Logical Record Activity Descriptions</td>
<td>LRACT</td>
<td>Reports on program activity for logical records defined in a specified subschema</td>
</tr>
<tr>
<td>Subschema Logical Record Descriptions</td>
<td>LRDEFS</td>
<td>Reports on logical records defined in a specified subschema</td>
</tr>
<tr>
<td>Full Report Name</td>
<td>Syntax Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Logical Record Path Descriptions</td>
<td>LRPATH</td>
<td>Reports on the paths defined for logical records in a specified subschema</td>
</tr>
<tr>
<td>Subschema Area Description Listing</td>
<td>SUBAREA</td>
<td>Reports on areas copied into a specified subschema</td>
</tr>
<tr>
<td>Subschema Record Description Listing</td>
<td>SUBREC</td>
<td>Provides comprehensive information on records copied into a specified subschema</td>
</tr>
<tr>
<td>Subschema Set Description Listing</td>
<td>SUBSET</td>
<td>Reports on sets copied into a specified subschema</td>
</tr>
</tbody>
</table>

**Physical Database Definition reports**

Physical database definition reports provide information on specified DMCLs, SEGMENTS, and DBTABLES defined in the dictionary.

<table>
<thead>
<tr>
<th>Full Report Name</th>
<th>Syntax Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical Database Report Listing</td>
<td>PDBRPT</td>
<td>Produces all physical database definition reports</td>
</tr>
<tr>
<td>DBTABLE Listing</td>
<td>DBTLST</td>
<td>Reports on the database names defined in the specified database names table</td>
</tr>
<tr>
<td>DMCL Listing</td>
<td>DMCLST</td>
<td>Reports on files, segments, and areas defined in the specified DMCL</td>
</tr>
<tr>
<td>Segment Listing</td>
<td>SEGLST</td>
<td>Reports on files, areas, and symbolics defined in the specified segment</td>
</tr>
</tbody>
</table>

**Authorization**

*Physical database definition reports*

<table>
<thead>
<tr>
<th>To</th>
<th>You Need This Privilege or Authority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run DMCLST report</td>
<td>DBADMIN on the dictionary being processed or DISPLAY on each DMCL on which you want to report</td>
</tr>
<tr>
<td>Run SEGGLST reports</td>
<td>DBADMIN on the dictionary being processed or DISPLAY on each segment on which you want to report</td>
</tr>
<tr>
<td>DBTLST</td>
<td>DBADMIN on the dictionary being processed or DISPLAY on each DBTABLE on which you want to report</td>
</tr>
</tbody>
</table>

*Non-database, schema, and subschema reports*

<table>
<thead>
<tr>
<th>To</th>
<th>You Need This Privilege On Dictionary being processed if SECURITY FOR IDD SIGNON IS ON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run any reports except physical database definition reports</td>
<td>SIGNON authority</td>
</tr>
</tbody>
</table>
To You Need This Privilege

Run all global reports except physical database definition reports DISPLAY or higher Basic entity type on which you want to report if SECURITY FOR IDD SIGNON IS ON

Run schema and subschema reports DISPLAY or higher Database entity on which you want to report if SECURITY FOR IDMS IS ON

Run all reports except User Listing and physical database definition reports DISPLAY or higher Entity occurrence on which you want to report if the PUBLIC ACCESS IS NONE

Syntax

SIGNON USER statement

```
SIGNON USER user-id PASSWORD = password
```

SCHEMA statement

```
SCHEMA schema-name
  ( schema-name )
ALL

Version version-number
  ( version-number )
HIGHEST
ALL
```

SUBSCHEMA statement

```
SUBSCHEMA subschema-name
  ( subschema-name )
ALL
```

DMCL statement

```
DMCL dmcl-name
  ( dmcl-name )
ALL
```

SEGMENT statement

```
SEGMENT segment-name
  ( segment-name )
ALL
```
 Statements

 Statement descriptions

 IDMSRPTS utility processing is controlled by the input statements listed in the following table.

⚠️ Note: When specifying multiple input statements (other than SIGNON and REPORTS SELECTION), you must specify them in the order listed in the following table.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIGNON USER</td>
<td>Identifies the user running IDMSRPTS.</td>
</tr>
<tr>
<td>SCHEMA</td>
<td>Specifies the schema(s) to report on.</td>
</tr>
<tr>
<td>SUBSCHEMA</td>
<td>Specifies the subschema(s) to report on.</td>
</tr>
<tr>
<td>DMCL</td>
<td>Specifies the DMCL(s) to report on.</td>
</tr>
<tr>
<td>SEGMENT</td>
<td>Specifies the segment(s) to report on.</td>
</tr>
</tbody>
</table>
DBTABLE  Specifies the database name table(s) to report on.
REPORTS SELECTION  Specifies the report(s) to generate.

SIGNON USER statement

The SIGNON USER statement is provided to override the external user ID for IDD security checking at IDD signon and for entity type and entity occurrence levels for reports other than physical database definition reports.

\[\textbf{Note:} Your external user ID (from the job card) is always used for physical database definition security checking.\]

If you do not specify a SIGNON USER statement, your external user ID is also used for IDD security checking.

Use the SIGNON USER statement, specifying the user ID and password of an authorized user, if your external user ID is:

- Not defined in the dictionary and IDD SIGNON IS ON
- Is defined in the dictionary but does not have authority to sign on or access the entity-types and occurrences on which you wish to report

If the IDD USER SIGNON OVERRIDE option does not allow signon override, and the external user ID is different from the SIGNON user ID, CA IDMS/DB ignores the SIGNON user ID for all IDD security checking.

If the dictionary is unsecured, you do not need to specify either a user ID or a password.

Syntax

\[
\text{SIGNON USER user-id [P@Sword = password]}
\]

Parameters

- **user-id**
  Identifies a user in the dictionary.

- **PASsword = password**
  Specifies the password assigned to the user in the data dictionary.
  A password must be provided if a SIGNON statement has been processed, and the specified user ID is not the same as the external user ID and has been assigned a password in the dictionary.

SCHEMA statement

The SCHEMA statement identifies the schema(s) to report on.
A schema statement is required if the reports selection includes any of the following:

- SCHRPT
- AREALST
- PGMLIST
- RECDES
- SETDES

Authorization

The identified user must hold authority for:

- IDD SIGNON if IDD SIGNON is secured and
- DISPLAY or higher if IDMS is secured and
- DISPLAY or higher on named schema(s) if PUBLIC ACCESS for the occurrence is NONE

Syntax

```
SCHema  [ = ]  schema-name
         ( [ = ]  schema-name )
         ALL

Version  [ = ]  version-number
         ( [ = ]  version-number )
         HIGhest
         ALL
```

Parameters

- **SCHema**
  Identifies the schema(s) for which specified schema-level reports are to be produced, and/or the schema(s) to which any specified subschema(s) are related.

- **schema-name**
  Specifies the name of a schema defined in the data dictionary. You can specify multiple occurrences of `schema-name` by enclosing them in parentheses and separating them with commas.
  IDMSRPTS reports only on the named schema(s).
  You can specify up to 25 schema names.

- **ALL**
  Directs IDMSRPTS to report on all schemas in the data dictionary.

- **Version**
  Specifies version numbers for all the schemas named in the SCHEMA parameter.
- **version-number**
  Directs IDMSRPTS to report on the specified versions of each schema. You can specify up to 25 version numbers.

- **HIGhest**
  Directs IDMSRPTS to report on the highest numbered version of each specified schema. This is the default.

- **ALL**
  Directs IDMSRPTS to report on all versions of each specified schema.

**SUBSCHEMA statement**

The SUBSCHEMA statement identifies the subschema(s) to report on.

Both a schema and subschema statement are required if the reports selection includes any of the following:

- SSCRPT
- DATDIR
- LRACT
- LRDEFS
- LRPATH
- SUBREC
- SUBSET

**Authorization**

The identified user must hold authority for:

- IDD SIGNON if IDD SIGNON is secured and
- DISPLAY or higher if IDMS is secured and
- DISPLAY or higher on requested schema(s) if PUBLIC ACCESS for the occurrence is NONE

**Syntax**

```
SUBSCHEMA SSCC = subschema-name
                   ( ALL subschema-name )
```

**Parameters**
- **SUBschema/SSC**
  Identifies the subschema(s) for which the specified subschema-level reports are to be produced. SUBSCHEMA and SSC are synonyms and can be used interchangeably.

- **subschema-name**
  Specifies the name of a subschema compiled under a schema identified by the SCHEMA statement. IDMSRPTS reports only on the named subschema(s).
  You can specify up to 25 subschema names.
  You can specify multiple occurrences of *subschema-name* by enclosing them in parentheses and separating them with commas.

- **ALL**
  Directs IDMSRPTS to reports on all subschemas compiled under the schema(s) identified by the SCHEMA statement.

**DMCL statement**

The DMCL statement identifies the DMCL(s) to report on.

A DMCL statement is required if the reports selection includes DMCLST.

**Authorization**

The external user ID must hold either DBADMIN authority for the dictionary or DISPLAY authority on requested DMCL(s).

**Syntax**

```plaintext
►─┬────────────────────────────────────────────┬─────────────────────────────►
└─ DMCl ─┬─────┬─┬─ dmcl-name ──────────────┬─┘
    ┌───── ─┘     ┌───── ─┐     │
    └─ ( ─▼─ dmcl-name ─┴─ ) ─┤
        └─ ALL ────────────┘
```

**Parameters**

- **DMCl**
  Identifies the DMCL(s) for which the DMCLST report is to be produced.

- **dmcl-name**
  Identifies the name of a DMCL defined in the catalog component of the dictionary. You can specify multiple occurrences of *dmcl-name* by enclosing them in parentheses and separating them with commas.
  IDMSRPTS reports only on the named DMCL(s).
  You can specify up to 25 DMCL names.

- **ALL**
  Directs IDMSRPTS to reports on all DMCLs defined in the dictionary.

**SEGMENT statement**

The SEGMENT statement identifies the segment(s) to report on.
A SEGMENT statement is required if the reports selection includes SEGLST.

**Authorization**

The external user ID must hold either DBADMIN authority for the dictionary or DISPLAY authority on requested segment(s).

**Syntax**

```
 SEGMENT segment-name
   ( segment-name )
   ALL
```

**Parameters**

- **SEGMENT**
  Identifies the segment(s) for which the SEGLST report is to be produced.

- **segment-name**
  Specifies the name of a segment defined in the catalog component of the dictionary. You can specify multiple occurrences of `segment-name` by enclosing them in parentheses and separating them with commas.
  IDMSRPTS reports only on the named segment(s).
  You can specify up to 25 segment names.

- **ALL**
  Directs IDMSRPTS to reports on all segments defined in the dictionary.

**DBTABLE statement**

The DBTABLE statement identifies the database name table(s) to report on.

A DBTABLE statement is required if the reports selection includes DBTLST.

**Authorization**

The external user ID must hold either DBADMIN authority for the dictionary or DISPLAY authority on requested database names table(s).

**Syntax**

```
 DBTable dbtable-name
   ( dbtable-name )
   ALL
```

**Parameters**

- **DBTable**
  Identifies the database name table(s) for which specified dbtable-level reports are to be produced.
- **dbtable-name**
  Identifies the name of a database name table defined in the catalog component of the dictionary. You can specify multiple occurrences of `dbtable-name` by enclosing them in parentheses and separating them with commas. IDMSRPTS reports only on the named database name table(s).

- **ALL**
  Directs IDMSRPTS to report on all database name tables defined in the catalog segment of the dictionary.

**REPORTS SELECTION statement**

The REPORTS SELECTION statement identifies the reports to generate.

**Syntax**

```
   ┌─┬───────────────────────┬──────────────────────────────────────────────────┐◄
   │ │  ALLRPT  ──┬─┘
   │ └─▼─┬─ AREALST ──┴─┘
   │ │ DATDIR ─┘
   │ │ DBTLST ─┘
   │ │ DMCLST ─┘
   │ │ GLBLRPT ─┘
   │ │ LRACT ─┘
   │ │ LRDEFS ─┘
   │ │ LRPATH ─┘
   │ │ MODLST ─┘
   │ │ MODULE ─┘
   │ │ PDBRPT ─┘
   │ │ PGMLST ─┘
   │ │ PROCESS ─┘
   │ │ PRTLST ─┘
   │ │ QFILE ─┘
   │ │ RECCOPY ─┘
   │ │ RECDES ─┘
   │ │ SCHRPT ─┘
   │ │ SEGLST ─┘
   │ │ SETDES ─┘
   │ │ SSCRPT ─┘
   │ │ SUBAREA ─┘
   │ │ SUBREC ─┘
   │ │ SUBSET ─┘
   │ │ TABLE ─┘
   │ └─▼─┬─ USER ──┴─┘
   └─▼─┘
```

**Parameters**

- **ALLRPT**
  Generates all non-database reports; all reports except DBTLST, DMCLST, and SEGLST.

- **AREALST**
  Generates the Area Listing for each specified schema.

- **DATDIR**
  Generates the Subschema Data Directory Listing for each specified subschema.

- **DBTLST**
  Generates the DBNAMES Table Report.

- **DMCLST**
  Generates the DMCL Listing for each specified DMCL.
GLBLRPT
Generates all database independent reports: MODULE, PROCESS, TABLE, RECCOPY, PRTLST, QFILE, and USER.
No SCHEMA, SUBSCHEMA, DMCL, or SEGMENT statement is required to generate these reports.

LRACT
Generates the Logical Record Activity Descriptions report for each specified subschema.

LRDEFS
Generates the Subschema Logical Record Descriptions report for each specified subschema.

LRPATH
Generates the Logical Record Path Descriptions report for each specified subschema.

MODLST
Generates the module, process, and table reports.

MODULE
Generates the Module Description Listing, which lists each module that you are authorized to display. Processes, tables, qfiles, and protocols are not included in this report.

PDBRPT
Generates the physical database reports: DBTLST, DMCLST, and SEGLST.
If specified:
- DMCLST will be produced for all DMCLs identified on DMCL statements
- SEGLST will be produced for all SEGMENTs identified on SEGMENT statements
- DBTLST will be produced for all DBTABLES identified on DBTABLE statements

PGMLST
Generates the Program Cross-Reference Listing for each specified schema version.

PROCESS
Generates the ADS Process Description Listing, which lists each process that you are authorized to display.

PRTLST
Generates the Protocol Listing.

QFILE
Generates the OLQ Qfile Description Listing.

RECCOPY
Generates the Record Copy Description Listing, which lists each IDD-built record, report, and transaction that you are authorized to display.

RECDES
Generates the Schema Record Description Listing for each specified schema version.
• **SCHRPT**
  Generates all schema reports for each specified schema version: AREALST, PGMLST, RECDES, and SETDES.
  The schema reports do not report on unvalidated schemas or schemas that have errors.

• **SEGLST**
  Generates the Segment Listing for each specified segment.

• **SETDES**
  Generates the Set Description Listing for each specified schema version.

• **SSCRPT**
  Generates all subschema reports for each specified subschema, schema, and version: DATDIR, LRACT, LRDEFS, LRPATH, SUBREC, and SUBSET.
  The subschema reports do not report on unvalidated subschemas or subschemas that contain errors.

• **SUBAREA**
  Generates the Subschema Area Description Listing for each specified subschema.

• **SUBREC**
  Generates the Subschema Record Description Listing for each specified subschema.

• **SUBSET**
  Generates the Subschema Set Description Listing for each specified subschema.

• **TABLE**
  Generates the Table Description Listing, which lists each table that you are authorized to display.

• **USER**
  Generates the User Listing.

---

**Usage**

**Input**

Input to the IDMSRPTS utility consists of statements to control the utility processing.

⚠️ **Note:** When specifying multiple input statements (other than SIGNON and REPORTS SELECTION), you must specify them in this order:

1. SCHEMA
2. SUBSCHEMA
3. DMCL
4. SEGMENT

5. DBTABLE

**Specifying a dictionary name and nodename**

To override the default dictionary name or dictionary nodename the IDMSRPTS utility uses to produce reports, use the DICTNAME and DICTNODE parameters in the SYSIDMS parameter file.

**Note:** For a complete description of the SYSIDMS parameter file, see the *CA IDMS Common Facilities Section*.

**Output**

The IDMSRPTS utility generates reports on information stored in a data dictionary.

**Batch operating mode**

You can execute the IDMSRPTS utility either in local mode or under the central version.

**JCL Considerations**

For more information about the JCL used to execute IDMSRPTS, see the section for your operating system in this section.

**Examples**

**Protocol Listing**

The following input parameters direct IDMSRPTS to generate the Protocol Listing.

`prtlst`

**DMCL, SEGMENT, DBTABLE Listing**

The following input parameters direct IDMSRPTS to generate the DMCL Listing for the DMCL IDMSDMCL, a Segment Listing for each segment defined in the dictionary and a DBTABLE Listing for each DBTABLE defined in the dictionary.

`dmcl=idmsdmcl`  
`segment=all`  
`dbtable=all`  
`pdrpt`

**Schema Record and Set Description Listing**

The following input parameters direct IDMSRPTS to generate the Schema Record Description Listing and the Set Description Listing for versions 1, 99, and 100 of the schema EMPSCHM.
Subschema Area and Data Directory Listing

The following input parameters direct IDMSRPTS to generate the Subschema Area Description Listing and the Subschema Data Directory Listing for each subschema compiled under the highest existing version of the schema EMPSCHM.

```
schema=empschm subschema=all
subarea datadir
```

Sample Output

### Protocol Listing

```
GLOBAL
IDMSRPTS mm.n
PRTLST

------------ PROTOCOL LISTING ------------
<table>
<thead>
<tr>
<th>PROTOCOL NAME</th>
<th>PROTOCOL VERSION</th>
<th>DATE CREATED</th>
<th>LAST UPDATED</th>
<th>LANGUAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRTLST</td>
<td>0001</td>
<td>mm/dd/yy</td>
<td>N/A</td>
<td>COBOL</td>
</tr>
</tbody>
</table>

MOVE @DMLSEQ TO DML-SEQUENCE
MOVE @OCCUR TO RECORD-OCCUR
MOVE @NODN TO SSC-NODN
MOVE @DBN TO SSC-DBN
MOVE @DICTNO TO SSC-DNO
MOVE @DICTNA TO SSC-DNA
MOVE @LNG TO LRC-LRPXELNG
MOVE @LRSIZE TO LRC-MAXVXP
MOVE '@LRSTAT' TO LR-STATUS
MOVE '@VERB' TO LRVERB
MOVE '@NAME' TO LRNAME

PALOOP
    MOVE @PXE TO @PXELNG
ADD @PXELNG TO LRC-LRPXELNG

PALOOP
    CALL 'IDMS' USING SUBSCHEMA-CTRL @CALLIDMS
IDBMSCOM (@FUNC1)
    @SSCTRL
    @SSNAME
    @SSDCTRL
    @REC
    @SET
    @AREA
    @DATANAME
    @RECBUF
    @OCCUR2
IDBMSCOM (@FUNC2)
IDBMSCOM (@KEEP)
    @P1
    @P2
    @P3
    @P4

PALOOP
    @PA1
    @PA2
    @PA3

PALOOP
    IF ERROR-STATUS EQUAL TO '@STATUS'
```

...
### DMCL, SEGMENT, and DBTABLE Listing

**IDMSRPTS nn.n**  
**----------- DMCL LISTING -----------**  
**DATE** **TIME** **PAGE**

**DMCLST**  
DICTIONARY SYSTEM OF NODE DEFAULT

**SEGLST**  
DICTIONARY SYSTEM OF NODE DEFAULT

<table>
<thead>
<tr>
<th>EXTERNAL NAME</th>
<th>LOW BLOCK NR</th>
<th>HIGH BLOCK NR</th>
<th>PAGE GROUP</th>
<th>LOW PAGE NUMBER</th>
<th>CALC HIGH PAGE NUMBER</th>
<th>HIGH PAGE NUMBER</th>
<th>BLK/PAGE</th>
<th>PAGE</th>
</tr>
</thead>
</table>

**DMCL CVDMCL13**

**EXTERNAL** **LOW** **REL** **HIGH** **REL** **PAGE** **LOW** **PAGE** **CALC** **HIGH** **HIGH** **PAGE** **BLK*/PAGE** **PAGE**

**NAME** **BLOCK NR** **BLOCK NR** **GROUP** **NUMBER** **PAGE NR** **NUMBER** **SIZE** **RESV**

<table>
<thead>
<tr>
<th>SEGMENT ...............</th>
<th>APPLCAT</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE ..................</td>
<td>APPLCAT</td>
<td>APPLCAT</td>
</tr>
<tr>
<td>AREA ..................</td>
<td>DDLCAT</td>
<td>1</td>
</tr>
<tr>
<td>FILE ..................</td>
<td>APPLCATX</td>
<td>APPLCATX</td>
</tr>
<tr>
<td>AREA ..................</td>
<td>DDLCATX</td>
<td>1</td>
</tr>
<tr>
<td>FILE ..................</td>
<td>APPLCATL</td>
<td>APPLCATL</td>
</tr>
<tr>
<td>AREA ..................</td>
<td>DDLCATL0D</td>
<td>1</td>
</tr>
</tbody>
</table>

**SEGMENT ............... | APPLNMK | 0 |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE ..................</td>
<td>APPLML</td>
<td>APPLML</td>
</tr>
<tr>
<td>AREA ..................</td>
<td>DDLML</td>
<td>1</td>
</tr>
<tr>
<td>FILE ..................</td>
<td>ASFDML</td>
<td>ASFDML</td>
</tr>
<tr>
<td>AREA ..................</td>
<td>DDLML</td>
<td>1</td>
</tr>
<tr>
<td>FILE ..................</td>
<td>ASFDNF</td>
<td>ASFDNF</td>
</tr>
<tr>
<td>AREA ..................</td>
<td>IDMSR-AREA</td>
<td>1</td>
</tr>
<tr>
<td>FILE ..................</td>
<td>ASFDATA</td>
<td>ASFDATA</td>
</tr>
<tr>
<td>AREA ..................</td>
<td>IDMSR-AREA2</td>
<td>1</td>
</tr>
</tbody>
</table>

**SEGMENT ............... | CATSYS | 0 |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE ..................</td>
<td>DCAT</td>
<td>DCCAT</td>
</tr>
</tbody>
</table>

**IDMSRPTS nn.n**  
**----------- SEGMENT LISTING -----------**  
**DATE** **TIME** **PAGE**

**SEGLST**  
DICTIONARY SYSTEM OF NODE DEFAULT

**EXTERNAL NAME** **LOW** **REL** **HIGH** **REL** **PAGE GROUP** **LOW PAGE NUMBER** **CALC HIGH PAGE NUMBER** **HIGH PAGE NUMBER** **BLK/PAGE** **PAGE**

<table>
<thead>
<tr>
<th>SEGMENT ...............</th>
<th>APPLCAT</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>AREA ..................</td>
<td>DDLCAT</td>
<td>0</td>
</tr>
<tr>
<td>WITHIN FILE ....</td>
<td>APPLCAT</td>
<td>APPLCAT</td>
</tr>
<tr>
<td>AREA ..................</td>
<td>DDLCATX</td>
<td>0</td>
</tr>
<tr>
<td>WITHIN FILE ....</td>
<td>APPLCATX</td>
<td>APPLCATX</td>
</tr>
<tr>
<td>AREA ..................</td>
<td>DDLCATL0D</td>
<td>0</td>
</tr>
<tr>
<td>WITHIN FILE ....</td>
<td>APPLCATL</td>
<td>APPLCATL</td>
</tr>
</tbody>
</table>

**IDMSRPTS nn.n**  
**----------- DBTABLE LISTING -----------**  
**DATE** **TIME** **PAGE**

**DBTLSST**  
DICTIONARY SYSTEM OF NODE DEFAULT

**EXTERNAL NAME** **LOW** **REL** **HIGH** **REL** **PAGE GROUP** **LOW PAGE NUMBER** **CALC HIGH PAGE NUMBER** **HIGH PAGE NUMBER** **BLK/PAGE** **PAGE**

<table>
<thead>
<tr>
<th>SEGMENT ...............</th>
<th>APPLCAT</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>AREA ..................</td>
<td>DDLCAT</td>
<td>0</td>
</tr>
<tr>
<td>WITHIN FILE ....</td>
<td>APPLCAT</td>
<td>APPLCAT</td>
</tr>
<tr>
<td>AREA ..................</td>
<td>DDLCATX</td>
<td>0</td>
</tr>
<tr>
<td>WITHIN FILE ....</td>
<td>APPLCATX</td>
<td>APPLCATX</td>
</tr>
<tr>
<td>AREA ..................</td>
<td>DDLCATL0D</td>
<td>0</td>
</tr>
<tr>
<td>WITHIN FILE ....</td>
<td>APPLCATL</td>
<td>APPLCATL</td>
</tr>
</tbody>
</table>
DBCFILE ........ CVDDBTB13

DBTABLE MAPPINGS

<table>
<thead>
<tr>
<th>Subschema IDMSCAT?</th>
<th>Maps To</th>
<th>Subschema IDMSCAT?</th>
<th>Using DBNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBSCHEMA IDMSRSSA</td>
<td>MAPS TO</td>
<td>SUBSCHEMA IDMSRSSA</td>
<td>USING DBNAME SYSTEM</td>
</tr>
<tr>
<td>SUBSCHEMA RC??????</td>
<td>MAPS TO</td>
<td>SUBSCHEMA RC??????</td>
<td>USING DBNAME ASFDICT</td>
</tr>
<tr>
<td>SUBSCHEMA RU??????</td>
<td>MAPS TO</td>
<td>SUBSCHEMA RU??????</td>
<td>USING DBNAME ASFDICT</td>
</tr>
<tr>
<td>SUBSCHEMA CITSALLP</td>
<td>MAPS TO</td>
<td>SUBSCHEMA CITSALLP</td>
<td>USING DBNAME INFODB</td>
</tr>
<tr>
<td>SUBSCHEMA MTSSINFPP</td>
<td>MAPS TO</td>
<td>SUBSCHEMA MTSSINFPP</td>
<td>USING DBNAME INFODB</td>
</tr>
</tbody>
</table>

DBNAME ...... APPLDICT MATCH ON SUBSCHEMA OPTIONAL

<table>
<thead>
<tr>
<th>Segment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPLCAT</td>
<td></td>
</tr>
<tr>
<td>APPLNWK</td>
<td></td>
</tr>
<tr>
<td>SYSMSG</td>
<td></td>
</tr>
</tbody>
</table>

DBNAME ...... ASFDICT MATCH ON SUBSCHEMA OPTIONAL

<table>
<thead>
<tr>
<th>Segment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASFNWK</td>
<td></td>
</tr>
<tr>
<td>SYSMSG</td>
<td></td>
</tr>
</tbody>
</table>

DBNAME ...... DIRLDICT MATCH ON SUBSCHEMA OPTIONAL

<table>
<thead>
<tr>
<th>Segment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIRLNWK</td>
<td></td>
</tr>
<tr>
<td>SYSMSG</td>
<td></td>
</tr>
</tbody>
</table>

DBNAME ...... INFODB MATCH ON SUBSCHEMA OPTIONAL

<table>
<thead>
<tr>
<th>Segment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFOTSIS</td>
<td></td>
</tr>
<tr>
<td>TSIS</td>
<td></td>
</tr>
</tbody>
</table>

DBNAME ...... INFODICT MATCH ON SUBSCHEMA OPTIONAL

<table>
<thead>
<tr>
<th>Segment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFONWK</td>
<td></td>
</tr>
</tbody>
</table>

DBNAME ...... SYSTEM MATCH ON SUBSCHEMA OPTIONAL

<table>
<thead>
<tr>
<th>Segment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATSYS</td>
<td></td>
</tr>
<tr>
<td>SYMSMG</td>
<td></td>
</tr>
<tr>
<td>SYSTEM</td>
<td></td>
</tr>
</tbody>
</table>

---

Schema Record and Set Description Listing

<table>
<thead>
<tr>
<th>IDMSRPTS nm.n</th>
<th>Schema Record Description Listing</th>
<th>Date</th>
<th>Time</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECORD NAME... COVERAGE</td>
<td>DICTIONARY APPLDICT OF NODE DEFAULT</td>
<td>mm/dd/yy</td>
<td>181426</td>
<td>1</td>
</tr>
<tr>
<td>RECORD VERSION... 0001</td>
<td>DLGTH= 36</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECORD ID......... 0400</td>
<td>KGTH= 28</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECORD LENGTH..... FIXED</td>
<td>DSTRT= 20</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

LOCATION MODE...... VIA SET EMP-COVERAGE DISPLACEMENT 0000 PAGES

<table>
<thead>
<tr>
<th>Data Item</th>
<th>Redefines</th>
<th>Usage</th>
<th>Value</th>
<th>Picture</th>
<th>Picture</th>
<th>Strt</th>
<th>Lght</th>
</tr>
</thead>
<tbody>
<tr>
<td>02 SELECTION-DATE-0400</td>
<td>DISPLAY</td>
<td>1</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>03 SELECTION-YEAR-0400</td>
<td>DISPLAY</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>03 SELECTION-MONTH-0400</td>
<td>DISPLAY</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>03 SELECTION-DAY-0400</td>
<td>DISPLAY</td>
<td>9</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01 TERMINATION-MONTH-0400</td>
<td>DISPLAY</td>
<td>9</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01 TERMINATION-DAY-0400</td>
<td>DISPLAY</td>
<td>9</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02 TYPE-0400</td>
<td>DISPLAY</td>
<td>X</td>
<td>13</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>88 MASTER-0400</td>
<td>COND</td>
<td>'M'</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>88 FAMILY-0400</td>
<td>COND</td>
<td>'F'</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>88 DEPENDENT-0400</td>
<td>COND</td>
<td>'D'</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02 INS-PLAN-CODE-0400</td>
<td>DISPLAY</td>
<td>X(3)</td>
<td>14</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>88 GROUP-LIFE-0400</td>
<td>COND</td>
<td>'001'</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>88 MMD-0400</td>
<td>COND</td>
<td>'002'</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Subschema Area and Data Directory Listing**

<table>
<thead>
<tr>
<th>RECORD: COVERAGE</th>
<th>ID: 0400 VER: 100 TYPE: I LEN: 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA NAME</td>
<td>LEVEL</td>
</tr>
<tr>
<td>SELECTION-DATE-0400</td>
<td>02</td>
</tr>
<tr>
<td>SELECTION-YEAR-0400</td>
<td>03</td>
</tr>
<tr>
<td>SELECTION-MONTH-0400</td>
<td>03</td>
</tr>
<tr>
<td>SELECTION-DAY-0400</td>
<td>03</td>
</tr>
<tr>
<td>TERMINATION-DATE-0400</td>
<td>02</td>
</tr>
<tr>
<td>TERMINATION-YEAR-0400</td>
<td>03</td>
</tr>
<tr>
<td>TERMINATION-MONTH-0400</td>
<td>03</td>
</tr>
<tr>
<td>TERMINATION-DAY-0400</td>
<td>03</td>
</tr>
<tr>
<td>TYPE-0400</td>
<td>02</td>
</tr>
<tr>
<td>MASTER-0400</td>
<td>88</td>
</tr>
<tr>
<td>FAMILY-0400</td>
<td>88</td>
</tr>
<tr>
<td>DEPENDENT-0400</td>
<td>88</td>
</tr>
<tr>
<td>INS-PLAN-CODE-0400</td>
<td>02</td>
</tr>
<tr>
<td>GROUP-LIFE-0400</td>
<td>88</td>
</tr>
<tr>
<td>HMO-0400</td>
<td>88</td>
</tr>
<tr>
<td>GROUP-HEALTH-0400</td>
<td>88</td>
</tr>
<tr>
<td>GROUP-DENTAL-0400</td>
<td>88</td>
</tr>
</tbody>
</table>

**Subschema Area and Data Directory Listing**

<table>
<thead>
<tr>
<th>RECORD: COVERAGE</th>
<th>ID: 0400 VER: 100 TYPE: I LEN: 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA NAME</td>
<td>LEVEL</td>
</tr>
<tr>
<td>COVSELDT</td>
<td>02</td>
</tr>
<tr>
<td>COVSELYR</td>
<td>03</td>
</tr>
<tr>
<td>COVSELMO</td>
<td>03</td>
</tr>
<tr>
<td>COVSELDA</td>
<td>03</td>
</tr>
<tr>
<td>COVTMMDT</td>
<td>02</td>
</tr>
<tr>
<td>COVTMMYR</td>
<td>03</td>
</tr>
<tr>
<td>COVTMMMO</td>
<td>03</td>
</tr>
<tr>
<td>COVTMMDA</td>
<td>03</td>
</tr>
<tr>
<td>COTYPE</td>
<td>02</td>
</tr>
<tr>
<td>COVMSTR</td>
<td>88</td>
</tr>
<tr>
<td>COVFAMILY</td>
<td>88</td>
</tr>
<tr>
<td>COVPMDT</td>
<td>88</td>
</tr>
<tr>
<td>COVPUCOC</td>
<td>02</td>
</tr>
<tr>
<td>GROUP-LIFE</td>
<td>88</td>
</tr>
<tr>
<td>HMO</td>
<td>88</td>
</tr>
<tr>
<td>GROUP-HEALTH</td>
<td>88</td>
</tr>
<tr>
<td>GROUP-DENTAL</td>
<td>88</td>
</tr>
</tbody>
</table>
IDMSRSTC

The IDMSRSTC schema compare utility generates IDMSRSTT macro statements for use in a database restructure operation. IDMSRSTC generates the statements by comparing two schemas:

- **An old schema** that describes the database before restructuring
- **A new schema** that describes the database after restructuring

The utility reads the schema definitions from the data dictionary.

This article describes the following information:

- Syntax (see page 349)
- Input Parameter Statements (see page 350)
- Usage (see page 352)
- Example (see page 353)
- Sample Output (see page 354)

**Syntax**

**SIGNON statement**

```
SIGNON

USER name = user-id
Password = password

.
```

**SCHEMA statement**

```
SCHEMA

OLD Schema name is old-schema-name

Version is
Highest
Lowest

NEW Schema name is new-schema-name

Version is
Highest
Lowest

.
```

**SIGNOFF statement**

```
SIGNOFF

BYE
LOGOFF

.
```
Input Parameter Statements

Parameter statement descriptions

IDMSRSTC utility processing is controlled by the following input parameter statements:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIGNON</td>
<td>Initiates IDMSRSTC processing</td>
</tr>
<tr>
<td>SCHEMA</td>
<td>Identifies the old and new schemas to be compared</td>
</tr>
<tr>
<td>SIGNOFF</td>
<td>Terminates IDMSRSTC processing</td>
</tr>
</tbody>
</table>

Coding considerations

You must code the IDMSRSTC input parameter statements in uppercase between columns 1 and 72, inclusive. Each statement must end with a period.

SIGNON statement

Must be the first parameter statement you submit to IDMSRSTC.

SCHEMA statement

You can include any number of SCHEMA statements.

SIGNOFF statement

The SIGNOFF statement must be the last parameter statement.

SIGNON statement

The SIGNON statement initiates IDMSRSTC processing and optionally specifies a user identifier and password.

Syntax

```
SIGnon

USEr name is/= user-id
PASsword is/= password

.
```

Parameters

- **USEr name is/= user-id**
  
  Identifies a user defined in the user catalog. The specified user must have the authority to access the schemas named in the IDMSRSTC run.
  
  If no USER ID is specified, USER ID is the user known to the execution environment. If SIGNON OVERRIDE is not allowed in the dictionary, USER ID, if specified, must be the same as that known to the execution environment.
  
  IS and = are synonyms and can be used interchangeably.
• **PASsword is/= password**
  Specifies the password associated with the user identified in the USER parameter. If the password contains embedded blanks, you must enclose it in site-standard quotation marks.
  You must include the PASSWORD parameter if the user specified in the USER parameter is associated with a password in the dictionary, unless the USER ID is the same as the user known to the execution environment. In this case, the password is ignored.
  IS and = are synonyms and can be used interchangeably.

**SCHEMA statement**

The SCHEMA statement identifies an old schema and a new schema to be compared by IDMSRSTC. For each SCHEMA statement you submit, IDMSRSTC generates a set of IDMSRSTT macro statements. Each set of macro statements begins with IDMSRSTT BUFSIZE and ends with END.

**Syntax**

```plaintext
OLD SCHEMA name is old-schema-name
  Version is version-number
    HIGhest
    LOWest

NEW SCHEMA name is new-schema-name
  Version is version-number
    HIGhest
    LOWest

.
```

**Parameters**

• **OLD SCHEMA name is old-schema-name**
  Specifies the name of a schema that describes the database before restructuring. *Old-schema-name* must identify a schema defined in the data dictionary.

• **Version is**
  Qualifies the old or new schema name with a version number.
  If you omit the VERSION parameter for the old or new schema, IDMSRSTC uses the dictionary default version number for existing entities.

• **version-number**
  Explicitly specifies the schema version number.

• **HIGhest**
  Directs IDMSRSTC to use the highest version number assigned to the named schema.

• **LOWest**
  Directs IDMSRSTC to use the lowest version number assigned to the named schema.

• **NEW SCHEMA name is new-schema-name**
  Specifies the name of a schema that describes the database after restructuring. *New-schema-name* must identify a schema defined in the data dictionary.

**SIGNOFF statement**

The SIGNOFF statement terminates IDMSRSTC processing.
SIGNOFF, BYE, and LOGOFF are synonyms and can be used interchangeably.

Syntax

![Syntax Diagram]

Usage

Input

Input to the IDMSRSTC utility consists only of statements to control the utility processing.

Output

The IDMSRSTC utility generates:

- A card-image file containing the IDMSRSTT macro statements with comments and error messages
- A formatted listing that duplicates the information in the card-image file

Execution mode

You can execute the IDMSRSTC utility either in local mode or under the central version.

Note: For more information about the JCL used to execute IDMSRSTC, see the section for your operating system in this section.

Specifying dictionary name and nodename

To override the default dictionary name or dictionary nodename that the IDMSRSTC utility will access, use the SYSIDMS DICTNAME and DICTNODE parameters. For a complete description of the SYSIDMS parameter file, see the CA IDMS Common Facilities Section.

Review the IDMSRSTT macro statements

You should review the IDMSRSTT macro statements generated by IDMSRSTC before assembling them into a base restructuring table. Informational and warning messages included in the IDMSRSTC listing indicate statements that might require modification. You will also need to modify the macro statements under the following circumstances:

<table>
<thead>
<tr>
<th>Circumstance</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Name the procedure in the NUPROCS parameter of the appropriate IDMSRSTT RECNAME statement.</td>
</tr>
<tr>
<td>Circumstance</td>
<td>Modification</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>A database procedure not included in the new schema is to be executed</td>
<td>Delete the IDMSRSTT FIELD statements for the redefining elements.</td>
</tr>
<tr>
<td>during the run of the RESTRUCTURE utility.</td>
<td>Alternatively, if only the prefix portion of the record is being restructured, and the record</td>
</tr>
<tr>
<td>A record being restructured has a field that is being expanded by increasing the number of positions to the right of the decimal point.</td>
<td>if the field is an unsigned zone. The new position in each statement should be changed to reflect the proper position in the record's new layout. Signed numeric or packed fields cannot be restructured by a single run of the RESTRUCTURE utility. To properly handle the sign nibble may require that the new bytes be placed to the left of the original field. You should run a user program to modify each affected record by multiplying the expanded field by the proper factor to realign the decimal position.</td>
</tr>
</tbody>
</table>

**Note:** For a description of the IDMSRSTT macro statements, see Section B, IDMSRSTT Macro Statements (see page 478).

**Assemble the IDMSRSTT macro statements**

After reviewing and modifying the IDMSRSTT macro statements, you must assemble the statements into a base restructuring table for use by the RESTRUCTURE and RESTRUCTURE CONNECT utilities.

**Multiple sets of IDMSRSTT macro statements**

IDMSRSTC can generate multiple sets of IDMSRSTT macro statements; one set for each SCHEMA statement you submit. RESTRUCTURE and RESTRUCTURE CONNECT, however, can use only one base restructuring table at a time. Therefore, you must assemble separately each set of IDMSRSTT macro statements generated by IDMSRSTC.

**Example**

If you execute IDMSRSTC with the input statements shown next, the utility generates IDMSRSTT macro statements that describe the differences between the LRDKSCHM and LRDKSCH2 schemas with the dictionary default version number.

```
signon usage mode is retrieval.
old schema name is lrdkschm
    new schema name is lrdksch2.
signoff.
```
Sample Output

The IDMSRSTC utility generates the following report after executing the parameters in the previous example.

<table>
<thead>
<tr>
<th>IDMSRSTC nn.n</th>
<th>CA</th>
<th>DATE</th>
<th>TIME</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>volser</td>
<td>RESTRUCTURE SCHEMA COMPARE ACTIVITY LIST</td>
<td>mm/dd/yy</td>
<td>16411366</td>
<td>0001</td>
</tr>
</tbody>
</table>

000001  SIGNON USAGE MODE IS RETRIEVAL.
000002  OLD SCHEMA NAME IS LRDKSCHM.
000003  NEW SCHEMA NAME IS LRDKSCH2.
000004  SIGNON USAGE MODE IS RETRIEVAL.
000004  SIGNOFF ACCEPTED WORD 2

** TRANSACTION SUMMARY **

<table>
<thead>
<tr>
<th>ENTITY</th>
<th>ADD</th>
<th>MODIFY</th>
<th>REPLACE</th>
<th>DELETE</th>
<th>DISPLAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

NO ERRORS OR WARNINGS ISSUED FOR THIS COMPILE

More Information

- For more information about the IDMSRSTT macro statements, see Section B, IDMSRSTT Macro Statements (see page 478).
- For more information about restructuring a database, see RESTRUCTURE (see page 241) and RESTRUCTURE CONNECT (see page 248).
z/OS JCL - IDMSBCF and Utilities

Sample z/OS JCL commands to run CA IDMS/DB utility statements and programs are presented in the following pages:

- **Batch Command Facility (z/OS) (see page 355)** - Presents common commands used to run the Batch Command Facility
- **Utility Statements (z/OS) (see page 357)** - Contains an alphabetical list of links to additional commands required to run each utility statement and program
- **The Utility Programs (z/OS) (see page 383)** - Present utility programs for the Batch Command Facility

### Batch Command Facility (z/OS)

The following z/OS JCL samples are used to execute the Batch Command Facility (IDMSBCF) for CA IDMS. When using the IDMSBCF program to execute a utility statement, code these statements along with the required statements for each of the utilities. The file assignments for each utility are presented on subsequent pages in this section. **Note:** For more information about the Command Facility, see the *CA IDMS Common Facilities Section*.

**Local mode IDMSBCF (z/OS)**

```
// EXEC PGM=IDMSBCF,REGION=2048K
//STEPLIB DD DSN=idsms.dba.loadlib,DISP=SHR
// DD DSN=idsms.custom.loadlib,DISP=SHR
// DD DSN=idsms.cagjload,DISP=SHR
//ddmsg DD DSN=idsms.sysmsg.ddldcmmsg,DISP=SHR
//sysjnl DD DSN=idsms.tapejnl,DISP=
//SYSLST DD SYSOUT=A
```

*Insert file assignments required by the utility statements*

```
//SYSIDMS DD *
```

*Insert SYSIDMS parameters if applicable*

```
//SYSIPT DD *
```

*Insert utility statements*

```
/*
```

**Note:** Additional file assignments might be needed for the user catalog and the system dictionary depending on your security implementation.

**Note:** DD statements for database and native VSAM files can be omitted if data set name information is specified in the file definitions.

<table>
<thead>
<tr>
<th>idms.dba.loadlib</th>
<th>Data set name of the load library containing the DMCL and database name table load modules</th>
</tr>
</thead>
</table>
CA IDMS Reference - 19.0

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>idms.custom.loadlib</td>
<td>Data set name of the load library containing customized CA IDMS system software modules</td>
</tr>
<tr>
<td>idms.cagjload</td>
<td>Data set name of the load library containing vanilla CA IDMS system software modules</td>
</tr>
<tr>
<td>dcmgs</td>
<td>DDname of the system message (DDLDCMSG) area</td>
</tr>
<tr>
<td>idms.sysmsg.ddldcmgs</td>
<td>Data set name of the system message (DDLDCMSG) area</td>
</tr>
<tr>
<td>sysjrnl</td>
<td>DDname of the tape journal file, if one is defined in the DMCL</td>
</tr>
<tr>
<td>idms.tapejrnl</td>
<td>Data set name of the tape journal file, if one is defined in the DMCL</td>
</tr>
<tr>
<td>SYSIDMS</td>
<td>DDname of the parameter file provided by CA IDMS to specify runtime directives and operating system-dependent parameters. Note: For a complete description of the SYSIDMS parameter file, see the CA IDMS Common Facilities Section.</td>
</tr>
</tbody>
</table>

Central version IDMSBCF (z/OS)

```
//STEPLIB      EXEC PGM=IDMSBCF,REGION=2048K
//STEPLIB      DD DSN=idms.dba.loadlib,DISP=SHR
//STEPLIB      DD DSN=idms.custom.loadlib,DISP=SHR
//STEPLIB      DD DSN=idms.cagjload,DISP=SHR
//SYSCTL       DD DSN=idms.sysctl,DISP=SHR
//dcmgs        DD DSN=idms.sysmsg.ddldcmgs,DISP=SHR
//sysjrnl      DD DUMMY
//SYSLST       DD SYSOUT=A

Insert file assignments required by the utility statements

SYSIDMS DD *
Insert SYSIDMS parameters as appropriate

SYSIPT DD *
Insert utility statements

* 
```

Note: The DD statement for the system message area (SYSMSG.DDLDCMSG) can be omitted if its data set name is specified in the DMC

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>idms.dba.loadlib</td>
<td>Data set name of the load library containing the DMCL and database name table load modules</td>
</tr>
<tr>
<td>idms.custom.loadlib</td>
<td>Data set name of the CA IDMS/DB load library containing customized load modules</td>
</tr>
<tr>
<td>idms.cagjload</td>
<td>Data set name of the CA IDMS/DB load library containing vanilla load modules</td>
</tr>
<tr>
<td>dcmgs</td>
<td>DDname of the system message (DDLDCMSG) area</td>
</tr>
<tr>
<td>idms.sysmsg.ddldcmgs</td>
<td>Data set name of the system message (DDLDCMSG) area</td>
</tr>
<tr>
<td>sysjrnl</td>
<td>DDname of the tape journal file, if one is defined in the DMCL</td>
</tr>
<tr>
<td>SYSIDMS</td>
<td></td>
</tr>
</tbody>
</table>
DDname of the parameter file provided by CA IDMS to specify runtime directives and operating system-dependent parameters.

**Note:** For a complete description of the SYSIDMS parameter file, see the *CA IDMS Common Facilities Section*.

## Utility Statements (z/OS)

This section lists the following utility statements:

- ARCHIVE JOURNAL (see page 358)
- ARCHIVE LOG (see page 358)
- ARCHIVE TRACE (see page 359)
- BACKUP (see page 359)
- BUILD (see page 359)
- CLEANUP (see page 360)
- CONVERT CATALOG (see page 361)
- CONVERT PAGE (see page 361)
- EXPAND PAGE (see page 362)
- EXTRACT JOURNAL (see page 362)
- FASTLOAD (see page 363)
- FIX ARCHIVE (see page 364)
- FIX PAGE (see page 365)
- FORMAT (see page 365)
- INSTALL STAMPS (see page 366)
- LOAD (see page 367)
- LOCK (see page 368)
- MAINTAIN INDEX (see page 368)
- MERGE ARCHIVE (see page 370)
- PRINT INDEX (see page 370)
- PRINT JOURNAL (see page 370)
- PRINT LOG (see page 371)
- PRINT PAGE (see page 371)
- PRINT SPACE (see page 372)
  - PRINT TRACE (see page 372)
- PUNCH (see page 372)
- RELOAD (see page 373)
- REORG (see page 375)
- RESTORE (see page 377)
- RESTRUCTURE (see page 377)
- RESTRUCTURE CONNECT (see page 378)
- ROLLBACK (see page 378)
- ROLLFORWARD (see page 378)
- SYNCHRONIZE STAMPS (see page 379)
ARCHIVE JOURNAL

DD statements for the batch command facility (z/OS)

```
//j1jrnl DD DSN=idms.j1jrnl,DISP=SHR
//j2jrnl DD DSN=idms.j2jrnl,DISP=SHR
Additional journal file assignments, as required

//archjrn1 DD DSN=idms.archive,DISP=(NEW,KEEP),
  //UNIT=tape,VOL=SER=nnnnnn/*NOTE: The blksize and lrecl are set by the
  /* The file record format is fixed.
Additional archive journal file assignments, as required

j1jrnl DDname of the first disk journal file, as defined in the DMCL
idms.j1jrnl Data set name of the first disk journal file

j2jrnl DDname of the second disk journal file, as defined in the DMCL
idms.j2jrnl Data set name of the second disk journal file

archjrnl DDname of the tape archive file, as defined in the DMCL
idms.archive Data set name of the tape archive file
tape Symbolic device name of the tape archive file
nnnnnn Volume serial number of the tape archive file
```

ARCHIVE LOG

DD statements for the batch command facility (z/OS)

```
//dlogdb DD DSN=idms.dlogdb,DISP=SHR
//dmsgdb DD DSN=idms.dmsgdb,DISP=SHR
//sysjrnl DD DUMMY
//SYS002 DD DSN=idms.archive,DISP=(NEW,CATLG)
  //DCB=(RECFM=VB,LRECL=280,BLKSIZE=bbbb)

dlogdb DDname of the system log area
idms. dlogdb Data set name of the system log area
dmsgdb DDname of the system message area
idms. dmsgdb Data set name of the system message area
```
sysjrnl | DDname of the tape journal file defined in the DMCL module  
---|---  
idms. archive | Data set name of the archive log file  

**ARCHIVE TRACE**

The CA IDMS installation media contains the following sample JCL ([https://docops.ca.com/display/IDMS/System+Generation+Syntax+Diagram+Conventions#SystemGenerationSyntaxDiagramConventions-SampleJCL](https://docops.ca.com/display/IDMS/System+Generation+Syntax+Diagram+Conventions#SystemGenerationSyntaxDiagramConventions-SampleJCL)) (intended for demonstration purposes only):

- **ATRCJCL** -- Contains a skeleton of the JCL required to run the ARCHIVE TRACE utility.

**BACKUP**

**DD statements for the batch command facility (z/OS)**

```java
//userdb DD DSN=user.userdb,DISP=SHR
Additional file assignments, as required
//SYS001 DD DSN=user.bkpfile,DISP=(NEW,PASS),
//UNIT=tapeout,VOL=SER=nnnnnnn,
//DCB=(RECFM=VB,BLKSIZE=bbbb)
```

- **userdb** | DDname of the database file  
- **user.userdb** | Data set name of the database file  
- **user.bkpfile** | Data set name of the tape backup file  
- **tapeout** | Symbolic device name of the tape backup file  
- **nnnnnnn** | Volume serial number of the tape backup file  
- **bbbb** | Block size of the tape backup file. Must be as large as the smaller of:  
The size of the largest page being backed up, plus 8  
32,760

**BUILD**

**DD statements for the batch command facility (z/OS)**

```java
//userdb DD DSN=user.userdb,DISP=disp//SYS002 DD DSN=user.load,DISP=OLD
//SYS003 DD DSN=user.build,DISP=(NEW,PASS),UNIT=tape,
//DCB=(RECFM=VB,LRECL=rrr,BLKSIZE=bbbb)
//SYSPOCH DD DSN=6&sortbuild,DISP=(NEW,PASS),
//UNIT=disk,SPACE=(TRK,1),DCB=BLKSIZE=80
//SORTMSG DD SYSPUT=A,DCB=BLKSIZE=bbbb
//SORTWK01 DD UNIT=disk,SPACE=(TRK,(nnn,nnn))
```
The SORTMSG and SORTWKnn files are only needed when performing a complete BUILD.

Add additional SORTWKnn files as necessary.

<table>
<thead>
<tr>
<th>userdb</th>
<th>DName of the database file</th>
</tr>
</thead>
<tbody>
<tr>
<td>user.db</td>
<td>Data set name of the database file</td>
</tr>
<tr>
<td>user.load*</td>
<td>Data set name of the input (SYS002) file; for sizing information see discussion of SYS003 in LOAD (see page 109)</td>
</tr>
<tr>
<td>user.build*</td>
<td>Data set name of the output (SYS003) file; for sizing information see BUILD (see page 43)</td>
</tr>
<tr>
<td>tape</td>
<td>Symbolic device name of the SYSPCH file</td>
</tr>
<tr>
<td>rrr</td>
<td>Record size of the SYS003 file</td>
</tr>
<tr>
<td>bbb</td>
<td>Block size of the SYS003 file</td>
</tr>
<tr>
<td>&amp;sortbu</td>
<td>Data set name of the SYSPCH file</td>
</tr>
<tr>
<td>disk</td>
<td>Symbolic device name of the SYSPCH file</td>
</tr>
</tbody>
</table>

⚠️ **Note:** When running a complete BUILD, SYS002 and SYS003 must point to the same intermediate file. When running a stepped BUILD, SYS002 and SYS003 must point to a different intermediate file.

**CLEANUP**

**DD statements for the batch command facility (z/OS)**

```
//sysjrn1 DD DSN=idms.tapejrn1,DISP=SHR
//userdb DD DSN=user.userdb,DISP=SHR
```

Additional database file assignments, as required

<table>
<thead>
<tr>
<th>sysjrn1</th>
<th>DDname of the tape journal file</th>
</tr>
</thead>
<tbody>
<tr>
<td>idms.tapejrn1</td>
<td>Data set name of the tape journal file</td>
</tr>
<tr>
<td>userdb</td>
<td>DDname of the user database file</td>
</tr>
<tr>
<td>user.userdb</td>
<td>Data set name of the user database file</td>
</tr>
</tbody>
</table>

**Central version**

To execute CLEANUP SEGMENT under the central version, modify the JCL shown previously as follows:

- Optionally remove any journal and database DD statements.


- Insert the following statement:

  ```
  //SYSCTL  DD  DSN=idms.sysctl,DISP=SHR
  ```

  `sysctl`  DDname of the SYSCTL file

  `idms.sysctl`  Data set name of the SYSCTL file

---

**CONVERT CATALOG**

**DD statements for the batch command facility (z/OS)**

- ```
  //ddlcat  DD  DSN=ucat.ddlcat,DISP=OLD
  ```

  `ddlcat`  DDname of the database file containing the DDLCAT area of the dictionary to be converted

- ```
  //ddlcatx  DD  DSN=ucat.ddlcatx,DISP=OLD
  ```

  `ucat.ddlcatx`  Data set name of the database file containing the DDLCAT area of the dictionary to be converted

- ```
  //j1jrnl  DD  DSN=tape.j1jrnl,(NEW,disp)
  ```

  `j1jrnl`  DDname of the first journal file, as defined in the DMCL

  `tape.j1jrnl`  Data set name of the first journal file

  `disp`  Disposition of the first journal file

### Additional journal file assignments, as required.

---

**Central version**

To execute CONVERT CATALOG under the central version, modify the previous JCL as follows:

- Optionally remove any journal and dictionary DD statements.

- Insert the following statement:

  ```
  //SYSCTL  DD  DSN=idms.sysctl,DISP=SHR
  ```

  `sysctl`  DDname of the SYSCTL file

  `idms.sysctl`  Data set name of the SYSCTL file

---

**CONVERT PAGE**

**DD statements for the batch command facility (z/OS)**
//userdb  DD  DSN=\(user\).userdb,DISP=SHR
Additional database file DD statements as needed.
//newdb  DD  DSN=\(user\).newdb,DISP=SHR
Additional new converted database file DD statements as needed.

<table>
<thead>
<tr>
<th>DDname</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userdb</td>
<td>DDname of the input database file</td>
</tr>
<tr>
<td>user.userdb</td>
<td>Data set name of the input database file</td>
</tr>
<tr>
<td>newdb</td>
<td>DDname of the output converted database file</td>
</tr>
<tr>
<td>user.newdb</td>
<td>Data set name of the output converted database file</td>
</tr>
</tbody>
</table>

**EXPAND PAGE**

DD statements for the batch command facility (z/OS)

//userdb  DD  DSN=\(user\).userdb,DISP=OLD
Additional existing database file assignments, as required
//xfile  DD  DSN=\(user\).xbase,DISP=(NEW,CATLG),UNIT=disk,
//        VOL=SER=nnnnnnn,SPACE=(expanded-database-size)
Additional expanded database file assignments, as required

<table>
<thead>
<tr>
<th>DDname</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userdb</td>
<td>DDname of the existing database file (as specified by the FILE parameter)</td>
</tr>
<tr>
<td>user.userdb</td>
<td>Data set name of the existing database file</td>
</tr>
<tr>
<td>xfile</td>
<td>DDname of the expanded database file (as specified by the INTO parameter)</td>
</tr>
<tr>
<td>user.xbase</td>
<td>Data set name of the expanded database file</td>
</tr>
<tr>
<td>disk</td>
<td>Symbolic device name of the expanded database file</td>
</tr>
<tr>
<td>nnnnnnnn</td>
<td>Volume serial number of the expanded database file</td>
</tr>
<tr>
<td>expanded-database-size</td>
<td>Space allocation for the expanded database file</td>
</tr>
</tbody>
</table>

**EXTRACT JOURNAL**

DD statements for the batch command facility (z/OS)

//SYS001  DD  DSN=\&archive\.\,DISP=(OLD)
//extract  DD  DSN=jrnl.extract,DISP=(NEW,CATLG),
//          DCB=(RECFM=VB,BLKSIZE=bbbb),
//          UNIT=nnn, VOL=SER=nnnnnnn
//SORTMSG  DD  SYSOUT=A
//SORTWK01  DD  UNIT=disk,SPACE=(TRK,(nnn,nnn))
Add additional SORTWKnn files as necessary
Data set name of the complete archive or journal file. It may be on tape or disk and concatenated.

**extract**  DDname of the extract journal file. If you don’t specify, it defaults to SYS002.

**jrnl.**  Data set name for the extract journal file

**bbb**  Block size of the extract journal file. Specify a size at least as large as the largest block size on the journal or archive files being processed.

### FASTLOAD

**DD statements for the batch command facility (z/OS)**

```
//SYS001 DD DSN=&.&sortuml.d.,DISP=SHR
//SYS002 DD DSN=user.db1001,DISP=(OLD,DELETE)
//SYS004 DD DSN=user.db1004,DISP=(NEW,PASS),UNIT=tape4, 
  DCB=(RECFM=VB,LRECL=rrr004,BLKSIZE=bbb004)
//SYS005 DD DSN=user.db1005,DISP=(NEW,PASS),UNIT=tape5, 
  DCB=(RECFM=VB,LRECL=rrr005,BLKSIZE=bbb005)
//SYS009 DD DSN=user.db1009,DISP=(NEW,PASS),UNIT=tape9, 
  DCB=(RECFM=VB,LRECL=rrr009,BLKSIZE=bbb009)
//SYS010 DD DSN=user.db1010,DISP=(NEW,PASS),UNIT=tape10, 
  DCB=(RECFM=VB,LRECL=rrr010,BLKSIZE=bbb010)
//SYS011 DD DSN=user.db1011,DISP=(NEW,PASS),UNIT=tape11, 
  DCB=(RECFM=VB,LRECL=rrr011,BLKSIZE=bbb011)
//SYSPCH DD DSN=&.&sortl.d,DISP=(NEW,PASS),UNIT=disk, 
  SPACE=(TRK,1),DCB=BLKSIZE=80
//RELDCTL DD DSN=user.reldctl,DISP=SHR,UNIT=nnnn, 
  VOL=SER=nnnnnn, 
  DCB=(RECFM=FB,LRECL=60,BLKSIZE=bbbctl)
//SORTMSG DD SYSPUT=A,DCB=BLKSIZE=bbb 
//SORTWK01 DD UNIT=disk,SPACE=(TRK,(nnn,nnn))
```

Add additional SORTWKnn files as necessary

//userdb DD DSN=user.userdb,DISP=SHR

Additional existing database files assignments, as required
<table>
<thead>
<tr>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>user. dbi005</td>
<td>Data set name of the intermediate work file containing a control record and set membership information from IDMSDBL2</td>
</tr>
<tr>
<td>tape005</td>
<td>Symbolic device name of the SYS005 file</td>
</tr>
<tr>
<td>rrr005</td>
<td>Record size of the SYS005 file; for sizing information, see FASTLOAD (see page 75)</td>
</tr>
<tr>
<td>bbb005</td>
<td>Block size of the SYS005 intermediate work file; must be at least rrr005 plus four bytes</td>
</tr>
<tr>
<td>user. dbi009</td>
<td>Data set name of the intermediate work file containing the sorted contents of SYS005</td>
</tr>
<tr>
<td>tape009</td>
<td>Symbolic device name of the SYS009 file</td>
</tr>
<tr>
<td>rrr009</td>
<td>Record size of the SYS009 file; should be the same as rrr005</td>
</tr>
<tr>
<td>bbb009</td>
<td>Block size of the SYS009 intermediate work file; should be the same as bbb005</td>
</tr>
<tr>
<td>user. dbi010</td>
<td>Data set name of the intermediate work file containing pointer information from IDMSDBL3</td>
</tr>
<tr>
<td>tape010</td>
<td>Symbolic device name of the SYS010 file</td>
</tr>
<tr>
<td>rrr010</td>
<td>Record size of the SYS010 file; for sizing information, see FASTLOAD (see page 75)</td>
</tr>
<tr>
<td>bbb010</td>
<td>Block size of the SYS010 intermediate work file; must be at least 60 bytes (the size of the control record plus four bytes)</td>
</tr>
<tr>
<td>user. dbi011</td>
<td>Data set name of the intermediate work file containing sorted pointer information from SORT4</td>
</tr>
<tr>
<td>tape011</td>
<td>Symbolic device name of the SYS011 file</td>
</tr>
<tr>
<td>rrr011</td>
<td>Record size of the SYS011 file; should be the same as rrr010</td>
</tr>
<tr>
<td>bbb011</td>
<td>Block size of the SYS011 intermediate work file; should be the same as bbb010</td>
</tr>
<tr>
<td>&amp;&amp;sortld</td>
<td>Data set name of the SYSPCH file</td>
</tr>
<tr>
<td>disk</td>
<td>Symbolic device name of the SYSPCH file</td>
</tr>
<tr>
<td>user. reldctl</td>
<td>Data set name of the reload control file containing control and set information</td>
</tr>
<tr>
<td>bbbctl</td>
<td>Blocksize of the RELDCTL file. It should be a multiple of 60 with a maximum size of 32,760.</td>
</tr>
<tr>
<td>userdb</td>
<td>DDname of the database file</td>
</tr>
<tr>
<td>user. userdb</td>
<td>Data set name of the database file</td>
</tr>
</tbody>
</table>

**FIX ARCHIVE**

**DD statements for the batch command facility (z/OS)**

```plaintext
//SYS001 DD DSN=idsms.tjrnldold,DISP=(OLD,PASS),UNIT=tapein, //SYS002 DD DSN=idsms. tjrnlnfix,DISP=(NEW,PASS),UNIT=tapeout, // DCB=BLKSIZE=bbbb
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>idms.tjrnldold</td>
<td>Data set name of the input tape journal file</td>
</tr>
<tr>
<td>tapein</td>
<td>Symbolic device name of the input tape journal file</td>
</tr>
</tbody>
</table>
**FIX PAGE**

**DD statements for the batch command facility (z/OS)**

```plaintext
//userdb DD DSN=user.userdb,DISP=OLD
Additional file assignments, as required
```

<table>
<thead>
<tr>
<th>DDname of the user file</th>
<th>Data set name of the user file</th>
</tr>
</thead>
<tbody>
<tr>
<td>userdb</td>
<td>user.userdb</td>
</tr>
</tbody>
</table>

**Central version**

To execute FIX PAGE under the central version, modify the JCL shown previously as follows:

- Optionally remove any journal and database DD statements.
- Insert the following statement:

```plaintext
//SYSCTL DD DSN=idms.sysctl,DISP=SHR
```

<table>
<thead>
<tr>
<th>DDname of the SYSCTL file</th>
<th>Data set name of the SYSCTL file</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysctl</td>
<td>idms.sysctl</td>
</tr>
</tbody>
</table>

**FORMAT**

**DD statements for the batch command facility (z/OS)**

To format a new database file:

```plaintext
//userdb DD DSN=user.userdb,DISP=(NEW,disp),
// UNIT=disk,VOL=SER=nnnnnnn,
// SPACE=(space)
```

To reformat an existing database file:

```plaintext
//userdb DD DSN=user.userdb,DISP=(OLD,PASS)
```

To format a new disk journal file:

```plaintext
//j1jrnl DD DSN=idms.j1jrnl,DISP=(NEW,disp),
// UNIT=disk,VOL=SER=nnnnnn,
// SPACE=(space)
```
To reformat an existing disk journal file:

```
//j1jrnl  DD  DSN=idms.j1jrnl,DISP=(OLD,PASS)
```

Additional database and journal file assignments, as required

To format a new SYSTRK file:

```
//anydd  DD  DSN=user.sysstrkn,DISP=(NEW,disp),
//UNIT=disk,VOL=SER=nnnnnn,
//SPACE=(space)
```

To reformat an existing SYSTRK file:

```
//anydd  DD  DSN=user.sysstrkn,DISP=(OLD,PASS)
```

<table>
<thead>
<tr>
<th>userdb</th>
<th>DDname of the database file</th>
</tr>
</thead>
<tbody>
<tr>
<td>user.userdb</td>
<td>Data set name of the database file</td>
</tr>
<tr>
<td>disp</td>
<td>Disposition of the new file (CATLG, KEEP, or PASS)</td>
</tr>
<tr>
<td>disk</td>
<td>Symbolic device name of the file being formatted</td>
</tr>
<tr>
<td>nnnnnn</td>
<td>Volume serial number of the file being formatted</td>
</tr>
<tr>
<td>space</td>
<td>Space allocation for the file being formatted</td>
</tr>
<tr>
<td>j1jrnl</td>
<td>DDname of the disk journal file</td>
</tr>
<tr>
<td>idms.j1jrnl</td>
<td>Data set name of the disk journal file</td>
</tr>
<tr>
<td>anydd</td>
<td>Target DDname specified in the FORMAT SYSTRK statement</td>
</tr>
<tr>
<td>user.sysstrkn</td>
<td>Data set name of the SYSTRK file</td>
</tr>
</tbody>
</table>

Central version

To execute FORMAT AREA and FORMAT SEGMENT under the central version, modify the JCL shown previously as follows:

- Optionally remove any journal and database DD statements.
- Insert the following statement:

```
//SYSCTL  DD  DSN=idms.sysctl,DISP=SHR
```

<table>
<thead>
<tr>
<th>syscall</th>
<th>DDname of the SYSCTL file</th>
</tr>
</thead>
<tbody>
<tr>
<td>idms.sysctl</td>
<td>Data set name of the SYSCTL file</td>
</tr>
</tbody>
</table>

INSTALL STAMPS

Local mode DD statements for the batch command facility (z/OS)

```
//userdb  DD  DSN=user.userdb,DISP=OLD
//userdict DD  DSN=user.userdict,DISP=SHR
//sysjrn1  DD  DSN=idms.sysjrn1,DISP=OLD
```
Central version

To execute INSTALL STAMPS under the central version, modify the JCL shown previously as follows:

- Remove the USERDICT and SYSJRNL DD statements.
- Insert the following statement after the USERDB DD statement:
  
  ```
  //SYSCTL   DD  DSN=idms.sysctl,DISP=SHR
  ```

LOAD

**DD statements for the batch command facility (z/OS)**

```plaintext
//userdict DD DSN=user.userdict,DISP=disp
//SYS001 DD DSN=user.input,DISP=OLD
//SYS002 DD DSN=user.loadin,DISP=(NEW,PASS),UNIT=tapein,DCB=(RECFM=VB,LRECL=rrrin,BKSIZE=bbbin)
//SYS003 DD DSN=user.loadout,DISP=(NEW,PASS),UNIT=tapeout,DCB=(RECFM=VB,LRECL=rrrout,BLKSIZ=bbbout)
//SYSPCH DD DSN=sortload,DISP=(NEW,PASS),UNIT=disk,SPACE=(TRK,1),DCB=BLKSIZ=80
//SORTMSG DD SYSPCH A,DCB=BLKSIZ=bbbout
//SORTWK01 DD UNIT=disk,SPACE=(TRK,(nnn,nnn))
```

The SORTMSG and SORTWKnn files are only needed when performing a complete LOAD.
Add additional SORTWKnn files as necessary

```
```

---

**userdb**  DDname of the database file
**user.userdb**  Data set name of the database file

**userdict**  DDname of the database file containing the dictionary with the table definitions
**sysjrnl**  DDname of the tape journal file
**idms.sysjrnl**  Data set name of the tape journal file

**userdict**  DDname of the database file containing the dictionary with the table definitions
**user.userdict**  Data set name of the database file containing the dictionary with the table definitions

**userdb**  DDname of the database file being loaded
**user.userdb**  Data set name of the database file being loaded

**user.input**  Data set name of the input file

**user.loadin**  Data set name of the input SYS002 file; for sizing information see **LOAD** (see page 109)

**tapein**  Symbolic device name of the SYS002 file

**rrrin**  Record size of the SYS002 file
**bbbin**  Block size of the SYS002 file

**user.loadout**  * Data set name of the output SYS003 file; for sizing information see LOAD (see page 109)

**tapeout**  Symbolic device name of the SYS003 file

**rrrout**  Record size of the SYS003 file

**bbbout**  Block size of the SYS003 file

**&sortload**  Data set name of the SYSPCH file

**disk**  Symbolic device name of the SYSPCH file

**Note:** When running a complete LOAD, SYS002 and SYS003 must point to the same intermediate file. When running a stepped LOAD, SYS002 and SYS003 must point to a different intermediate file.

When running a complete LOAD, you must preallocate the file referenced by SYS002 and SYS003. Do not use a temporary data set for these files when running a complete LOAD.

---

**LOCK**

**DD statements for the batch command facility (z/OS)**

```plaintext
//userdb DD DSN=user.userdb,DISP=SHR
```

**userdb**  DDname of the database file

**user.userdb**  Data set name of the database file

**Central version**

To execute LOCK AREA under the central version, modify the JCL shown previously as follows:

- Optionally remove any journal and database DD statements.

- Insert the following statement:

```plaintext
//SYSCTL DD DSN=idms.sysctl,DISP=SHR
```

**sysctl**  DDname of the SYSCTL file

**idms.sysctl**  Data set name of the SYSCTL file

---

**MAINTAIN INDEX**

**DD statements for the batch command facility (z/OS)**

```plaintext
//userdb DD DSN=user.olddb,DISP=SHR
Additional existing database files assignments, as required
//SYS003 DD DSN=user.db1003,DISP=(NEW, PASS), UNIT=tape003,
DCB=(RECFM=VB, LRECL=rrr003, BLKSIZE=bb003)
```
Add additional SORTWKnn files as necessary

<table>
<thead>
<tr>
<th>userdb</th>
<th>DDname of the existing database file</th>
</tr>
</thead>
<tbody>
<tr>
<td>olddb</td>
<td>Data set name of the existing database file</td>
</tr>
<tr>
<td>tape003</td>
<td>Data set name of the intermediate work file containing index descriptors from IDMSTABX</td>
</tr>
<tr>
<td>rrr003</td>
<td>Record size of the SYS003 file; see MAINTAIN INDEX (see page 125) for sizing information.</td>
</tr>
<tr>
<td>bbb003</td>
<td>Block size of the SYS003 intermediate work file</td>
</tr>
<tr>
<td>tape004</td>
<td>Data set name of the intermediate work file containing the output from SORT3</td>
</tr>
<tr>
<td>rrr004</td>
<td>Record size of the SYS004 file; should be the same as rrr003</td>
</tr>
<tr>
<td>bbb004</td>
<td>Block size of the SYS004 intermediate work file; should be the same as bbb003</td>
</tr>
<tr>
<td>tape005</td>
<td>Data set name of the intermediate work file containing pointers for user owned index sets from IDMSDBL3</td>
</tr>
<tr>
<td>rrr005</td>
<td>Record size of the SYS005 file; see MAINTAIN INDEX (see page 125) for sizing information</td>
</tr>
<tr>
<td>bbb005</td>
<td>Block size of the SYS005 intermediate work file</td>
</tr>
<tr>
<td>tape006</td>
<td>Data set name of the intermediate work file containing sorted pointers for user owned index sets from SORT4</td>
</tr>
<tr>
<td>rrr006</td>
<td>Record size of the SYS006 file; should be the same as rrr005</td>
</tr>
<tr>
<td>bbb006</td>
<td>Block size of the SYS006 intermediate work file; should be the same as bbb005</td>
</tr>
<tr>
<td>&amp;&amp;sort</td>
<td>Data set name of the SYSPCH file containing sort parameters from IDMSTABX and IDMSDBL3</td>
</tr>
<tr>
<td>disk</td>
<td>Data set name of the SYSPCH file</td>
</tr>
<tr>
<td>user.</td>
<td>Data set name of the reload control file containing control and set information</td>
</tr>
<tr>
<td>relcntl</td>
<td></td>
</tr>
<tr>
<td>bbbctl</td>
<td>Block size of the RELDCTL file. It should be a multiple of 60 with a maximum size of 32,760.</td>
</tr>
</tbody>
</table>
**MERGE ARCHIVE**

**DD statements for the batch command facility (z/OS)**

```plaintext
//SYS001  DD  DSN=idms.tjrnlold,DISP=(OLD,PASS),UNIT=tapein
//SYS002  DD  DSN=idms.tjrnlfix,DISP=(NEW,PASS),UNIT=tapeout,
            DCB=BLKSIZE=bbbb
//JRNM01  DD  DSN=idms.tmrgold,DISP=(OLD,PASS),UNIT=tapein
//SORTMSG  DD  SYSOUT=A,DCB=BLKSIZE=bbbb
//SORTWK01 DD  UNIT=disk,SPACE=(TRK,(nnn,nnn))
```

Add additional SORTWKnn files as necessary

<table>
<thead>
<tr>
<th><strong>idms.tjrnlold</strong></th>
<th>Data set name of the input tape journal file</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tapein</strong></td>
<td>Symbolic device name of the input tape journal file</td>
</tr>
<tr>
<td><strong>idms.tjrnlfix</strong></td>
<td>Data set name of the output tape journal</td>
</tr>
<tr>
<td><strong>tapeout</strong></td>
<td>Symbolic device name of the output tape journal file</td>
</tr>
<tr>
<td><strong>bbbb</strong></td>
<td>Block size of the output tape journal file, as defined in the DMCL</td>
</tr>
<tr>
<td><strong>idms.tmrgold</strong></td>
<td>Data set name of the input tape merged file. If none exists yet, specify //JRNM01 DD DUMMY</td>
</tr>
</tbody>
</table>

⚠️ **Note:** If the concatenated files have different block sizes, you must specify the following DCB parameter on the first file in the concatenation list:

```
DCB=(RECFM=VB,BLKSIZE=nnnn)
```

where nnnn is greater than or equal to (4 + the largest block size of any file in the concatenation list).

**PRINT INDEX**

**DD statements for the batch command facility (z/OS)**

```plaintext
//sysjrn1  DD  DUMMY
//userdb   DD  DSN=user.userdb,DISP=SHR
```

<table>
<thead>
<tr>
<th><strong>sysjrn1</strong></th>
<th>DDname of the dummy journal file</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>userdb</strong></td>
<td>DDname of the database file</td>
</tr>
<tr>
<td><strong>user.userdb</strong></td>
<td>Data set name of the database file</td>
</tr>
</tbody>
</table>

**PRINT JOURNAL**

**DD statements for the batch command facility (z/OS)**

```plaintext
//SYS001  DD  DSN=idms.tjrnlold,DISP=(OLD,PASS),UNIT=tapein
//SYS002  DD  DSN=idms.tjrnlfix,DISP=(NEW,PASS),UNIT=tapeout,
            DCB=BLKSIZE=bbbb
//JRNM01  DD  DSN=idms.tmrgold,DISP=(OLD,PASS),UNIT=tapein
//SORTMSG  DD  SYSOUT=A,DCB=BLKSIZE=bbbb
//SORTWK01 DD  UNIT=disk,SPACE=(TRK,(nnn,nnn))
```

Add additional SORTWKnn files as necessary

<table>
<thead>
<tr>
<th><strong>idms.tjrnlold</strong></th>
<th>Data set name of the input tape journal file</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tapein</strong></td>
<td>Symbolic device name of the input tape journal file</td>
</tr>
<tr>
<td><strong>idms.tjrnlfix</strong></td>
<td>Data set name of the output tape journal</td>
</tr>
<tr>
<td><strong>tapeout</strong></td>
<td>Symbolic device name of the output tape journal file</td>
</tr>
<tr>
<td><strong>bbbb</strong></td>
<td>Block size of the output tape journal file, as defined in the DMCL</td>
</tr>
<tr>
<td><strong>idms.tmrgold</strong></td>
<td>Data set name of the input tape merged file. If none exists yet, specify //JRNM01 DD DUMMY</td>
</tr>
</tbody>
</table>

⚠️ **Note:** If the concatenated files have different block sizes, you must specify the following DCB parameter on the first file in the concatenation list:

```
DCB=(RECFM=VB,BLKSIZE=nnnn)
```

where nnnn is greater than or equal to (4 + the largest block size of any file in the concatenation list).
PRINT LOG

**DD statements for the batch command facility (z/OS)**

To print from the DDLDCLOG area:

```
//dlogdb DD DSN=idms.dlogdb,DISP=SHR
```

To print from the archive log file:

```
//SYS001 DD DSN=idms.archjrnl,DISP=OLD
//dmsgdb DD DSN=idms.dmsgdb, DISP=SHR
//sysjrnl DD DSN=DUMMY
```

- **dlogdb** DDname of the data dictionary log area
- **idms.dlogdb** Data set name of the data dictionary log area
- **idms.archjrnl** Data set name of the archive log file
- **dmsgdb** DDname of the data dictionary message area
- **idms.dmsgdb** Data set name of the data dictionary message area
- **sysjrnl** DDname of the tape journal file defined in the DMCL module

PRINT PAGE

**DD statements for the batch command facility (z/OS)**

```
//userdb1 DD DSN=user.userdb1,DISP=SHR
//userdb2 DD DSN=user.userdb2,DISP=SHR
```

*Additional database file assignments, as required*

- **userdb1** DDname of the first database file
- **user.userdb1** Data set name of the first database file
- **userdb2** DDname of the second database file
- **user.userdb2** Data set name of the second database file

**Central version**

To execute PRINT PAGE under the central version, modify the JCL shown previously as follows:

- Optionally remove any journal and database DD statements.
Insert the following statement:

```
//SYSCTL   DD DSN=idms.sysctl,DISP=SHR
```

<table>
<thead>
<tr>
<th><code>sysctl</code></th>
<th>DDname of the SYSCTL file</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>idms.sysctl</code></td>
<td>Data set name of the SYSCTL file</td>
</tr>
</tbody>
</table>

## PRINT SPACE

**DD statements for the batch command facility (z/OS)**

```
//userdb1   DD DSN=user.userdb1,DISP=SHR
//userdb2   DD DSN=user.userdb2,DISP=SHR
```

*Additional database file assignments, as required*

<table>
<thead>
<tr>
<th><code>userdb1</code></th>
<th>DDname of the first database file</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>user.userdb1</code></td>
<td>Data set name of the first database file</td>
</tr>
<tr>
<td><code>userdb2</code></td>
<td>DDname of the second database file</td>
</tr>
<tr>
<td><code>user.userdb2</code></td>
<td>Data set name of the second database file</td>
</tr>
</tbody>
</table>

**Central version**

To execute PRINT SPACE FOR AREA or PRINT SPACE FOR SEGMENT under the central version, modify the JCL shown previously as follows:

- Optionally remove any journal and database DD statements.
- Insert the following statement:

```
//SYSCTL   DD DSN=idms.sysctl,DISP=SHR
```

<table>
<thead>
<tr>
<th><code>sysctl</code></th>
<th>DDname of the SYSCTL file</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>idms.sysctl</code></td>
<td>Data set name of the SYSCTL file</td>
</tr>
</tbody>
</table>

## PRINT TRACE

The CA IDMS installation media contains the following sample JCL (https://docops.ca.com/display/IDMS/System+Generation+Syntax+Diagram+Conventions#SystemGenerationSyntaxDiagramConventions-SampleJCL) (intended for demonstration purposes only):

- `PRTCJCL` - Contains a skeleton of the JCL required to run the PRINT TRACE utility.

## PUNCH

**Local mode DD statements for the batch command facility (z/OS)**
To execute PUNCH under the central version, modify the JCL shown previously as follows:

- Remove the USERCAT, USERCATX, USERCATL, and SYSIRNL DD statements.
- Insert the following statement:

  ```
  //SYSCTL      DD  DSN=idms.sysctl,DISP=SHR
  ```

**RELOAD**

**DD statements for the batch command facility (z/OS)**

```plaintext
//SYS001  DD  DSN=\&sortunld,DISP=SHR
//SYS002  DD  DSN=\&user.db1002,DISP=(OLD,DELETE)
//SYS003  DD  DSN=\&user.db1003,DISP=(OLD,DELETE)
//SYS004  DD  DSN=\&user.db1004,DISP=(NEW,PASS),UNIT=tape004,
//DCB=(RECFM=VB,LRECL=rrr004,BLKSIZE=bbb004,)
//SYS005  DD  DSN=\&user.db1005,DISP=(NEW,PASS),UNIT=tape005,
//DCB=(RECFM=VB,LRECL=rrr005,BLKSIZE=bbb005)
//SYS006  DD  DSN=\&user.db1006,DISP=(NEW,PASS),UNIT=tape006,
//DCB=(RECFM=VB,LRECL=rrr006,BLKSIZE=bbb006)
//SYS007  DD  DSN=\&user.db1007,DISP=(NEW,PASS),UNIT=tape007,
//DCB=(RECFM=VB,LRECL=rrr007,BLKSIZE=bbb007)
//SYS008  DD  DSN=\&user.db1008,DISP=(NEW,PASS),UNIT=tape008,
//DCB=(RECFM=VB,LRECL=rrr008,BLKSIZE=bbb008)
//SYS009  DD  DSN=\&user.db1009,DISP=(NEW,PASS),UNIT=tape009,
//DCB=(RECFM=VB,LRECL=rrr009,BLKSIZE=bbb009)
//SYS010  DD  DSN=\&user.db1010,DISP=(NEW,PASS),UNIT=tape010,
//DCB=(RECFM=VB,LRECL=rrr010,BLKSIZE=bbb010)
//SYS011  DD  DSN=\&user.db1011,DISP=(NEW,PASS),UNIT=tape011,
//DCB=(RECFM=VB,LRECL=rrr011,BLKSIZE=bbb011)
//SYS012  DD  DSN=\&sysreld,DISP=(NEW,PASS),UNIT=disk,
//SPACE=(TRK,1),DCB=BLKSIZE=80
//RELDCTL   DD  DSN=\&relctl,DISP=SHR
//SORTMSG   DD  SYSOUT=A,DCB=BLKSIZE=bbbb
```
Add additional SORTWKnn files as necessary

```
//SORTWK01 DD UNIT=DISK,SPACE=(TRK,(nnn,nnn))
```

Additional existing database files assignments, as required

<table>
<thead>
<tr>
<th>&amp;sortu</th>
<th>Data set name of the sort parameters created by UNLOAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>db1002</td>
<td>Data set name of the file put out by UNLOAD as SYS002</td>
</tr>
<tr>
<td>db1003</td>
<td>Data set name of the file put out by UNLOAD as SYS003</td>
</tr>
<tr>
<td>db1004</td>
<td>Data set name of the intermediate work file containing the output from SORT1</td>
</tr>
<tr>
<td>tape004</td>
<td>Symbolic device name of the SYS004 file</td>
</tr>
<tr>
<td>rrr004</td>
<td>Record size of the SYS004 file; should be the same as the record size for SYS002 used by UNLOAD</td>
</tr>
<tr>
<td>bbb004</td>
<td>Block size of the SYS004 intermediate work file; should be the same as the block size for SYS002 used by UNLOAD</td>
</tr>
<tr>
<td>db1005</td>
<td>Data set name of the intermediate work file containing member descriptors for chained sets from IDMSDBL2</td>
</tr>
<tr>
<td>tape005</td>
<td>Symbolic device name of the SYS005 file</td>
</tr>
<tr>
<td>rrr005</td>
<td>Record size of the SYS005 file; for sizing information, see RELOAD (see page 187)</td>
</tr>
<tr>
<td>bbb005</td>
<td>Block size of the SYS005 intermediate work file; must be at least 48 bytes (the size of a member descriptor plus four bytes)</td>
</tr>
<tr>
<td>db1006</td>
<td>Data set name of the intermediate work file containing member descriptors for user owned index sets from IDMSDBL2</td>
</tr>
<tr>
<td>tape006</td>
<td>Symbolic device name of the SYS006 file</td>
</tr>
<tr>
<td>rrr006</td>
<td>Record size of the SYS006 file; for sizing information, see RELOAD (see page 187)</td>
</tr>
<tr>
<td>bbb006</td>
<td>Block size of the SYS006 intermediate work file; must be at least 48 bytes (the size of a member descriptor plus four bytes)</td>
</tr>
<tr>
<td>db1007</td>
<td>Data set name of the intermediate work file containing the output of SORT2</td>
</tr>
<tr>
<td>tape007</td>
<td>Symbolic device name of the SYS007 file</td>
</tr>
<tr>
<td>rrr007</td>
<td>Record size of the SYS007 file; should be the same as the larger of: the record size for SYS003 used by UNLOAD the record size for SYS006</td>
</tr>
<tr>
<td>bbb007</td>
<td>Block size of the SYS007 intermediate work file; must be at least as large as the larger of: the block size for SYS003 used by UNLOAD the block size for SYS006</td>
</tr>
<tr>
<td>db1008</td>
<td>Data set name of the intermediate work file containing the reformatted index information from IDMSDBLX</td>
</tr>
<tr>
<td>tape008</td>
<td>Symbolic device name of the SYS008 file</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>rrr008</td>
<td>Record size of the SYS008 file; should be the same as rrr007</td>
</tr>
<tr>
<td>bbb008</td>
<td>Block size of the SYS008 intermediate work file; should be the same as bbb007</td>
</tr>
<tr>
<td>user.</td>
<td>Data set name of the intermediate work file containing the sorted index set descriptors from SORT3</td>
</tr>
<tr>
<td>dbi009</td>
<td>Symbolic device name of the SYS009 file</td>
</tr>
<tr>
<td>tape009</td>
<td>Record size of the SYS009 file; should be the same as the larger of:</td>
</tr>
<tr>
<td></td>
<td>the record size for SYS005</td>
</tr>
<tr>
<td></td>
<td>the record size for SYS008</td>
</tr>
<tr>
<td>rrr009</td>
<td>Block size of the SYS009 intermediate work file; must be at least as large as the larger of:</td>
</tr>
<tr>
<td></td>
<td>the block size for SYS003 used by UNLOAD</td>
</tr>
<tr>
<td></td>
<td>the block size for SYS006</td>
</tr>
<tr>
<td>bbb009</td>
<td>Data set name of the intermediate work file containing prefix pointer information from IDMSDBL3</td>
</tr>
<tr>
<td>tape010</td>
<td>Symbolic device name of the SYS010 file</td>
</tr>
<tr>
<td>rrr010</td>
<td>Record size of the SYS010 file; for sizing information, see RELOAD (see page 187)</td>
</tr>
<tr>
<td>bbb010</td>
<td>Block size of the SYS010 intermediate work file; must be at least 44 bytes (the size of a pointer descriptor plus four bytes)</td>
</tr>
<tr>
<td>user.</td>
<td>Data set name of the intermediate work file containing sorted prefix pointer information from SORT4</td>
</tr>
<tr>
<td>dbi011</td>
<td>Symbolic device name of the intermediate work file containing prefix pointer information from SORT4</td>
</tr>
<tr>
<td>tape011</td>
<td>Symbolic device name of the SYS011 file</td>
</tr>
<tr>
<td>rrr011</td>
<td>Record size of the SYS011 file; should be the same as rrr010</td>
</tr>
<tr>
<td>bbb011</td>
<td>Block size of the SYS011 intermediate work file; should be the same as bbb010</td>
</tr>
<tr>
<td>&amp;&amp;sortreld</td>
<td>Data set name of the SYSPCH file</td>
</tr>
<tr>
<td>disk</td>
<td>Symbolic device name of the SYSPCH file</td>
</tr>
<tr>
<td>user.</td>
<td>Data set name of the reload control file containing control and set information</td>
</tr>
<tr>
<td>relctl</td>
<td>Symbolic device name of the SYSPCH file</td>
</tr>
<tr>
<td>bbbctl</td>
<td>Blocksize of the RELDCTL file. It must be a multiple of 60 with a maximum size of 32,760.</td>
</tr>
<tr>
<td>userdb</td>
<td>DDname of the database file</td>
</tr>
<tr>
<td>user.</td>
<td>DDname of the database file</td>
</tr>
</tbody>
</table>

### REORG

**DD statements for the batch command facility (z/OS)**

**When allocating the REORG control file or when running setup only**

```bash
//RORGCTL DD DSN=user.reldctl,DISP=(NEW,CATLG),
//         UNIT=disk,VOL=SER=nnnnnnn,
//         DCB=(RECFM=FB,LRECL=8192,BLKSIZE=8192)
```

**Manual creation of work files other than DBKEYS files**
//wxnnnnn DD DSN=user.workfile,DISP=(NEW,CATLG),
//                     UNIT=disk, VOL=SER=nnnnnn,
//                     DCB=(RECFM=VB,BLKSIZE=wrkbbb)

Manual creation of DBKEYS work files

//WDnnnnnn DD DSN=user.workfile,DISP=(NEW,CATLG),
//                     UNIT=disk, VOL=SER=nnnnnn,
//                     DCB=(RECFM=FB,LRECL=16,BLKSIZE=dbkbbb)

Manual references to work files

//wxnnnnn DD DSN=user.workfile,DISP=SHR

File assignments when running the unload phase

//RORGCTL DD DSN=user.reldctl,DISP=SHR
//RORGJCL DD DSN=user.jclfile,DISP=SHR
//unlddb DD DSN=user.unldfile,DISP=SHR

Additional database file assignments, as required if not using dynamic allocation

Manual references to work files if not using dynamic allocation

File assignments when running the reload phase

//RORGCTL DD DSN=user.reldctl,DISP=SHR
//RORGJCL DD DSN=user.jclfile,DISP=SHR
//relddb DD DSN=user.reldfile,DISP=SHR

Additional database file assignments, as required if not using dynamic allocation

Manual references to work files if not using dynamic allocation

//SORTMSG DD SYSOUT=A
//SORTWK01 DD UNIT=disk, SPACE=(CYL,(nnn,nnn))

Additional SORTWKnn files as necessary

---

<table>
<thead>
<tr>
<th>User. relctl</th>
<th>Data set name of the REORG control file containing control information</th>
</tr>
</thead>
<tbody>
<tr>
<td>user.jclfile</td>
<td>Data set name of the file containing JCL for automatic job submission</td>
</tr>
<tr>
<td>wxnnnnnn</td>
<td>DDname for a work file. It must match the name generated in the Unload/Reload Work File Summary report.</td>
</tr>
<tr>
<td>user.workfile</td>
<td>Data set name of a work file when sectionly allocating work files</td>
</tr>
<tr>
<td>wrkbbb</td>
<td>Blocksize for a variable blocked sequential work file</td>
</tr>
<tr>
<td>WDnnnnnn</td>
<td>DDname for a DBKEYS file. It must match the name generated in the Unload/Reload Work File Summary report.</td>
</tr>
<tr>
<td>dbkbbb</td>
<td>Blocksize for a fixed-blocked, sequential DBKEYS file. Must be a multiple of 16.</td>
</tr>
<tr>
<td>disk</td>
<td>Symbolic unit name for a device</td>
</tr>
<tr>
<td>nnnnnn</td>
<td>Volume serial number</td>
</tr>
<tr>
<td>unlddb</td>
<td>DDname of the existing database file</td>
</tr>
</tbody>
</table>
**RESTORE**

**DD statements for the batch command facility (z/OS)**

```
//userdb1  DD  DSN=user.userdb1,DISP=OLD
//userdb2  DD  DSN=user.userdb2,DISP=OLD
//SYS001   DD  DSN=user.bkp,DISP=SHR
```

*Additional database file assignments, as required*

<table>
<thead>
<tr>
<th>DDname</th>
<th>Data set name</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>userdb1</code></td>
<td><code>user.userdb1</code></td>
</tr>
<tr>
<td><code>userdb2</code></td>
<td><code>user.userdb2</code></td>
</tr>
<tr>
<td><code>user.bkp</code></td>
<td><code>user.bkp</code></td>
</tr>
</tbody>
</table>

**RESTRUCTURE**

**DD statements for the batch command facility (z/OS)**

```
//userdb1  DD  DSN=user.userdb1,DISP=SHR
//userdb2  DD  DSN=user.userdb2,DISP=SHR
//SYS001   DD  DSN=idms.spill,DISP=(NEW,PASS)
//      DCB=(RECFM=FB,BLKSIZE=bbbb)
```

*Additional database file assignments, as required*

<table>
<thead>
<tr>
<th>DDname</th>
<th>Data set name</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>userdb1</code></td>
<td><code>user.userdb1</code></td>
</tr>
<tr>
<td><code>userdb2</code></td>
<td><code>user.userdb2</code></td>
</tr>
<tr>
<td><code>idms.spill</code></td>
<td><code>idms.spill</code></td>
</tr>
</tbody>
</table>
| `bbbb`        | Block size of the spill file; it should be a multiple of 40 with a maximum size of 32,760.
RESTRUCTURE CONNECT

DD statements for the batch command facility (z/OS)

```
//userdb    DD  DSN=user.userdb,DISP=SHR
//SYS001    DD  DSN=idms.spill,DISP=OLD
//SORTMSG   DD  SYSOUT=A,DCB=BLKSIZE=bbbb
//SORTWK01  DD  UNIT=disk,SPACE=(TRK,(nnn,nnn))
```

Add additional SORTWKnn files as necessary

<table>
<thead>
<tr>
<th>DDname</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userdb</td>
<td>DDname of the database file</td>
</tr>
<tr>
<td>user.userdb</td>
<td>Data set name of the database file</td>
</tr>
<tr>
<td>idms.spill</td>
<td>Data set name of the spill file.</td>
</tr>
<tr>
<td>bbbb</td>
<td>Block size of the spill file; it should be a multiple of 40 with a maximum size of 32,760</td>
</tr>
</tbody>
</table>

ROLLBACK

DD statements for the batch command facility (z/OS)

```
//SYS001  DD  DSN=&&archive,DISP=OLD,UNIT=tape//userdb    DD  DSN=user.userdb,DISP=OLD
```

Additional database file assignments, as required

If using the SORT option add these statements:

```
//SORTMSG   DD  SYSOUT=A,DCB=BLKSIZE=bbbb
//SORTWK01  DD  UNIT=disk,SPACE=(TRK,(nnn,nnn))
```

Add additional SORTWKnn files as necessary

<table>
<thead>
<tr>
<th>DDname</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;&amp;archive</td>
<td>Data set name of the complete archive or tape journal file</td>
</tr>
<tr>
<td>tape</td>
<td>Symbolic device name of the archive or tape journal file</td>
</tr>
<tr>
<td>userdb</td>
<td>DDname of the user database file</td>
</tr>
<tr>
<td>user.userdb</td>
<td>Data set name of the user database file</td>
</tr>
</tbody>
</table>

ROLLFORWARD

DD statements for the batch command facility (z/OS)

```
//userdb    DD  DSN=user.userdb,DISP=OLD
```

Add database file assignments, as required

If recovering from a standard journal file:

```
//SYS001  DD  DSN=&&archive.,DISP=(OLD),UNIT=tape
```
If recovering from a journal extract file:

//extract DD DSN=jrnl.extract,DISP=OLD

If using the SORT option or processing a journal extract file, add these statements:

//SORTMSG DD SYOUT=A,DCB=BLKSIZE=bbbb
//SORTWK01 DD UNIT=disk,SPACE=(TRK,(nnn,nnn))

Add SORTWKnn files as necessary

<table>
<thead>
<tr>
<th>userdb</th>
<th>DDname of the user database file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>user.userdb</td>
<td>Data set name of the user database file.</td>
</tr>
<tr>
<td>tape</td>
<td>Symbolic device name of the archive or tape journal file.</td>
</tr>
<tr>
<td>extract</td>
<td>DDname of the extract journal file. If not specified in the command it will default to SYS002.</td>
</tr>
<tr>
<td>jrn1.extract</td>
<td>Data set name of the extract journal file.</td>
</tr>
</tbody>
</table>

**SYNCHRONIZE STAMPS**

Local mode DD statements for the batch command facility (z/OS)

//userdb DD DSN=user.userdb,DISP=OLD
//userdict DD DSN=user.userdict,DISP=SHR
//sysjrnl DD DSN=idms.sysjrnl,DISP=OLD

<table>
<thead>
<tr>
<th>userdb</th>
<th>DDname of the database file</th>
</tr>
</thead>
<tbody>
<tr>
<td>user.userdb</td>
<td>Data set name of the database file</td>
</tr>
<tr>
<td>userdict</td>
<td>DDname of the database file containing the dictionary with the table definitions</td>
</tr>
<tr>
<td>sysjrnl</td>
<td>DDname of the tape journal file</td>
</tr>
<tr>
<td>idms.sysjrnl</td>
<td>Data set name of the tape journal file</td>
</tr>
</tbody>
</table>

Central version

To execute SYNCHRONIZE STAMPS under the central version, modify the JCL shown previously as follows:

- Remove the USERDICT and SYSJRNL DD statements.
- Insert the following statement after the USERDB DD statement:
  
  //SYSCtrl DD DSN=idms.sysctl,DISP=SHR

| idms.sysctl | Data set name of the SYSCtrl file |
TUNE INDEX

**DD statement for the batch command facility (z/OS)**

```
//userdb   DD   DSN=user.userdb,DISP=SHR
```

- **userdb** DDname of the database file
- **user.userdb** Data set name of the database file

**Central version**

To execute TUNE INDEX under the central version, modify the JCL shown previously as follows:

- Optionally remove any journal and database DD statements.
- Insert the following statement:
  ```
  //SYSCTL   DD   DSN=idms.sysctl,DISP=SHR
  ```

- **sysctl** DDname of the SYSCTL file
- **idms.sysctl** Data set name of the SYSCTL file

UNLOCK

**DD statements for the batch command facility (z/OS)**

```
//userdb   DD   DSN=user.userdb,DISP=SHR
```

- **userdb** DDname of the database file
- **user.userdb** Data set name of the database file

UNLOAD

**DD statements for the batch command facility (z/OS)**

```
//userdb   DD   DSN=user.olddb,DISP=SHR
```

**Additional existing database files assignments, as required**

- **//sysjrnld** DD DUMMY
- **//SYS002** DD DSN=user.db1002,DISP=(NEW,PASS),UNIT=tape002, DCB=(RECFM=VB,BLKSIZ=bbb002)
- **//SYS003** DD DSN=user.db1003,DISP=(NEW,PASS),UNIT=tape003, DCB=(RECFM=VB,BLKSIZ=bbb003)
- **//SYSPCH** DD DSN=&&sortunld,DISP=(NEW,PASS),UNIT=disk, SPACE=(TRK,1),DCB=BLKSIZ=80
- **//RELDCTL** DD DSN=user.reldctl,DISP=(NEW,CATLG),UNIT=nnnn, VOL=SER=nnnn, DCB=(RECFM=FB,LRECL=60,BLKSIZ=bbbctl)
<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userdb</td>
<td>DDname of the existing database file</td>
</tr>
<tr>
<td>user.olddb</td>
<td>Data set name of the existing database file</td>
</tr>
<tr>
<td>sysjrnl</td>
<td>DDname of the dummy journal file</td>
</tr>
<tr>
<td>user.db002</td>
<td>Data set name of the SYS002 output file</td>
</tr>
<tr>
<td>tape002</td>
<td>Symbolic device name of the SYS002 output file</td>
</tr>
<tr>
<td>bbb002</td>
<td>Block size of the SYS002 output file; must be at least the size of the largest database record plus 4 bytes</td>
</tr>
<tr>
<td>user.db003</td>
<td>Data set name of the SYS003 output file</td>
</tr>
<tr>
<td>tape003</td>
<td>Symbolic device name of the SYS003 output file</td>
</tr>
<tr>
<td>bbb003</td>
<td>Block size of the SYS003 output file</td>
</tr>
<tr>
<td>&amp;&amp;sortunl</td>
<td>Data set name of the sort parameters created by UNLOAD d</td>
</tr>
<tr>
<td>disk</td>
<td>Symbolic device name of the sort parameter file</td>
</tr>
<tr>
<td>user.relctl</td>
<td>Data set name of the reload control file containing control and set information</td>
</tr>
<tr>
<td>bbbctl</td>
<td>Size of the RELDCTL file. It should be a multiple of 60 with a maximum size of 32,760.</td>
</tr>
</tbody>
</table>

**UPDATE STATISTICS**

**DD statements for the batch command facility (z/OS)**

```plaintext
//userdb       DD DSN=user.userdb,DISP=SHR
Additional database file assignments, as required

//ddlcat      DD DSN=sysdict.ddlcat,DISP=OLD
//ddlxcat     DD DSN=sysdict.ddlxcat,DISP=OLD
//j1jrnl      DD DSN=tape.j1jrnl,DISP=(NEW,disp)
Additional journal file assignments, as required
```

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userdb</td>
<td>DDname of the database file</td>
</tr>
<tr>
<td>user.userdb</td>
<td>Data set name of the database file</td>
</tr>
<tr>
<td>ddcat</td>
<td>DDname of the database file containing the area of the system dictionary with the table definitions</td>
</tr>
<tr>
<td>sysdict.ddlcat</td>
<td>Data set name of the database file containing the area of the system dictionary with the table definitions</td>
</tr>
<tr>
<td>ddxcat</td>
<td>DDname of the database file containing the area of the system dictionary with indexes</td>
</tr>
<tr>
<td>sysdict.ddlxcat</td>
<td>Data set name of the database file containing the area of the system dictionary with indexes</td>
</tr>
<tr>
<td>j1jrnl</td>
<td>DDname of the first journal file, as defined in the DMCL</td>
</tr>
</tbody>
</table>
Central version

To execute UPDATE STATISTICS under the central version, modify the previous JCL as follows:

- Optionally remove any journal and database DD statements.
- Insert the following statement:

```jcl
//SYSCTL DD DSN=idsysctl,DISP=SHR
```

**VALIDATE**

**DD statements for the batch command facility (z/OS)**

```jcl
//userdb DD DSN=user.userdb,DISP=disp
//SYS002 DD DSN=user.valin,DISP=(NEW,PASS),UNIT=tapein, DCB=(RECFM=VB,LRECL=rrrin,BLKSIZE=bbbin)
//SYS003 DD DSN=user.valout,DISP=(NEW,PASS), UNIT=tapeout, DCB=(RECFM=VB,LRECL=rrrout, BLKSIZE=bbbout)
//SYSPCH DD DSN=&.&.sortval.,DISP=(NEW,PASS) UNIT=disk,SPACE=(TRK,1),DCB=BLKSIZE=80
//SORTMSG DD SYSOUT=A,DCB=BLKSIZE=bbbb
//SORTWK01 DD UNIT=disk,SPACE=(TRK,(nnn,nnn))
```

The SORTMSG and SORTWKnn files are only needed when performing a complete VALIDATE. Add additional SORTWKnn files as necessary.

**userdb**  DDname of the database file

**user.userdb**  Data set name of the database file

**user.valin**  Data set name of the input SYS002 file; for sizing information see VALIDATE (see page 295)

**tapein**  Symbolic device name of the SYS002 file

**rrrin**  Record size of the SYS002 file

**bbbin**  Block size of the SYS002 file

**user.valout**  Data set name of the output SYS003 file; for sizing information see VALIDATE (see page 295)

**tapeout**  Symbolic device name of the SYS003 file

**rrrout**  Record size of the SYS003 file

**bbbout**  Block size of the SYS003 file
Note: When running a complete VALIDATE, SYS002 and SYS003 must point to the same intermediate file. When running a stepped VALIDATE, SYS002 and SYS003 must point to a different intermediate file.

Utility Programs (z/OS)

This section lists the utility programs:

- IDMSDBAN (see page 383)
- IDMSDIRL (see page 384)
- IDMSLOOK (see page 385)
- IDMSRPTS (see page 386)
- IDMSRSTC (see page 387)
- IDMSRSTT (see page 388)

IDMSDBAN

Local mode IDMSDBAN (z/OS)

```
//DBN1 EXEC PGM=IDMSDBN1,REGION=region-size
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
// DD DSN=idms.custom.loadlib,DISP=SHR
//       DD DSN=idms.cagjload,DISP=SHR
//userdb DD DSN=user.userdb,DISP=SHR
```

Additional database file assignments, as required

```
//SYS002 DD DSN=&chain,DISP=(NEW,PASS),UNIT=tape,
// DCB=(RECFM=VB,BLKSIZE=9000)
//SYSLST DD SYSOUT=A
//SYSIPT DD *
```

Insert SYSIDMS parameters, as required

/*
IDMSDBAN input parameters
/*

```
//DBN2 EXEC PGM=IDMSDBN2,REGION=1500K
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
// DD DSN=idms.custom.loadlib,DISP=SHR
// DD DSN=idms.cagjload,DISP=SHR
//SYS001 DD DSN=&chain,DISP=(OLD),UNIT=tape
//SYS002 DD UNIT=SYSDA,SPACE=(TRK,(400,50)),
// DCB=(RECFM=VB,BLKSIZE=9000)
//SORTWK01 DD UNIT=SYSDA,SPACE=(TRK,(nnn,nn))
//SORTWK02 DD UNIT=SYSDA,SPACE=(TRK,(nnn,nn))
//SORTMSG DD SYSOUT=A
//SYSLST DD SYSOUT=A
```

Insert SYSIDMS parameters, as required
/*
Note: Additional file assignments may be needed for the user catalog and the system dictionary depending on your security implementation.

**region-size** The size of the region. This depends on the size of the database. A larger region size enables efficient execution of the internal sort.

**idms.dba.loadlib** Data set name of the load library containing the DMCL and database name table load modules

**idms.custom.loadlib** Data set name of the CA IDMS/DB load library containing customized load modules, including customized subschemas

**idms.cagjload** Data set name of the CA IDMS/DB load library containing vanilla load modules and delivered subschemas

**userdb** DDname of the user file

**user.userdb** Data set name of the user file

**&&chain** Data set name of the chain file generated by IDMSDBN1

**tape** Symbolic device name of the chain file

---

**IDMSDIRL**

**Local mode IDMSDIRL (z/OS)**

```bash
//DIRL EXEC PGM=IDMSDIRL,REGION=512K
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
// DD DSN=idms.custom.loadlib,DISP=SHR
// DD DSN=idms.cagjload,DISP=SHR
//sysjrnl DD DSN=idms.tapejrnl,DISP=(NEW,KEEP),UNIT=tape
//dictdb DD DSN=idms.dictdb,DISP=SHR
//SYSLST DD SYSOUT=A
//SYS001 DD DSN=idms.dirldata,DISP=(OLD,PASS),UNIT=tapein
//SYSIPT DD *
```

Note: Additional file assignments may be needed for the user catalog and the system dictionary depending on your security implementation.

**idms.dba.loadlib** Data set name of the load library containing the DMCL and database name table load modules

**idms.custom.loadlib** Data set name of the CA IDMS/DB load library containing customized load modules

**idms.cagjload** Data set name of the CA IDMS/DB load library containing vanilla load modules

**sysjrnl** DDname of the tape journal file

**idms.tapejrnl** Data set name of the tape journal file
Central version

To execute IDMSDIRL under the central version, modify the JCL shown previously as follows:

- Remove the SYSJRNL and DICTDB DD statements.
- Insert the following statement after the STEPLIB DD statement:

  ```
  //SYSCTL   DD   DSN=idms.sysctl,DISP=SHR
  ```

**idms.sysctl**  
Data set name of the SYSCTL file

⚠️ **Note:** An IDMSOPTI module link edited with IDMSDIRL can be used in place of or in addition to the SYSCTL file.

### IDMSLOOK

**Local mode IDMSLOOK (z/OS)**

```plaintext
//EXEC      PGM=IDMSLOOK,REGION=1024K
//STEPLIB    DD   DSN=idms.dba.loadlib,DISP=SHR
//            DD   DSN=idms.custom.loadlib,DISP=SHR
//            DD   DISP=idsms.cagjload,DISP=SHR
//sysjrnl    DD   DSN=idms.sysmsg.sysjrnl,DISP=SHR
//SYSLST     DD   SYSOUT=A
//SYSDMS     DD   *
```

Insert SYSIDMS parameters, as required

```plaintext
/*
//SYSIPT     DD   *
```

**IDMSLOOK input parameters**

⚠️ **Note:** Additional file assignments may be needed for the user catalog and the system dictionary depending on your security implementation.

**idms.dba.loadlib**  
Data set name of the load library containing the DMCL and database name table load modules
**IDMSRPTS**

**Local mode IDMSRPTS (z/OS)**

```jcl
//RPTS EXEC PGM=IDMSRPTS,REGION=256K
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
// DD DSN=idms.custom.loadlib,DISP=SHR
// DD DSN=idms.cagjload,DISP=SHR
//sysjrnl DD DSN=idms.tapejrnl,DISP=SHR
//dictdb DD DSN=idms.dictdb,DISP=SHR
//SYSOUT DD SYSOUT=A
//SYSLST DD SYSOUT=A
//SYSIDMS DD *

DICTNAME=dictionary-name
DICTNODE=node-name   This is optional

Insert other SYSIDMS parameters as appropriate

//SYSIPT DD *

Insert utility statements

/*

**Note:** Additional file assignments might be needed for the user catalog and the system dictionary depending on your security implementation.

---

**Central version**

To execute IDMSRPTS under the central version, modify the JCL shown previously as follows:
- Remove the SYSJRNL and DICTDB DD statements.

- Insert the following statement after the STEPLIB DD statement:

```
//SYSCTL DD DSN=idms.sysctl,DISP=SHR
```

<table>
<thead>
<tr>
<th><code>sysctl</code></th>
<th>DDname of the SYSCTL file</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>idms.sysctl</code></td>
<td>Data set name of the SYSCTL file</td>
</tr>
</tbody>
</table>

⚠️ **Note:** An IDMSOPTI module link edited with IDMSRPTS can be used in place of or in addition to the SYSCTL file.

### IDMSRSTC

**Local mode IDMSRSTC (z/OS)**

```
//RSTC EXEC PGM=IDMSRSTC,REGION=512K
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
// DD DSN=idms.custom.loadlib,DISP=SHR
// DD DSN=idms.cagjload,DISP=SHR
//SYSJRNL DD DSN=idms.tapejrnl,DISP=SHR
//DICTDD DD DSN=idms.dictdb,DISP=SHR
//SYSLST DD SYSOUT=A
//SYSPCH DD SYSOUT=B
//SYSIDMS DD *

DICTNAME=dictionary-name
DICTNODE=node-name

Insert other SYSIDMS parameters as appropriate

//SYSIPT DD *IDMSRSTC input parameters
/*
```

⚠️ **Note:** Additional file assignments might be needed for the user catalog and the system dictionary depending on your security implementation.

<table>
<thead>
<tr>
<th><code>idms.dba.loadlib</code></th>
<th>Data set name of the load library containing the DMCL and database name table load modules</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>idms.custom.loadlib</code></td>
<td>Data set name of the CA IDMS/DB load library containing customized load modules</td>
</tr>
<tr>
<td><code>idms.cagjload</code></td>
<td>Data set name of the CA IDMS/DB load library containing vanilla load modules</td>
</tr>
<tr>
<td><code>sysjrnl</code></td>
<td>DDname of the tape journal file</td>
</tr>
<tr>
<td><code>idms.tapejrnl</code></td>
<td>Data set name of the tape journal file</td>
</tr>
<tr>
<td><code>dictdb</code></td>
<td>DDname of the data dictionary file</td>
</tr>
</tbody>
</table>
Central version

To execute IDMSRSTC under the central version, modify the JCL shown previously as follows:

- Remove the SYSJRNL and DICTDB DD statements.
- Insert the following statement after the STEPLIB DD statement:
  ```
  //sysctl DD DSN=idms.sysctl,DISP=SHR
  ```

**Note:** An IDMSOPTI module link edited with IDMSRSTC can be used in place of or in addition to the SYSCTL file.

---

**IDMSRSTT**

**Assemble and link an IDMSRSTT module**

```hlasmcl
//ASMCL EXEC HLASMCL,
// PARM,C='OBJECT,NODECK,RENT'
//C.SYSLIB DD DISP=SHR,DSN=idms.cagjsrc
//C.SYSIN DD *

Put IDMSRSTT macro statements here

//L.SYSLMOD DD DISP=SHR,DSN=idms.custom.loadlib
//L.SYSIN DD *

Optionally add an INCLUDE statement for each procedure named in the NUPROCS= clause here. If not included, any procedures will be dynamically loaded.

ENTRY IDMSRSTT
NAME idmsrstt(R)
/*
```

**idms.cagjsrc** Library name containing source IDMS macros

**idms.custom.loadlib** Library name into which to save the new IDMSRSTT load module

**idmsrstt** Module name of the IDMSRSTT table
z/VSE JCL

This section includes sample z/VSE JCL to execute CA IDMS utilities. Additionally, a description of the =COPY facility, IDMSLBLS procedure, and the SYSIDMS parameter file, as they relate to the utilities, is included.

The files required to run each utility are identified in individual sections under, "JCL Considerations".

- =COPY Facility (see page 389)
- About the IDMSLBLS Procedure (see page 390)
- SYSIDMS Parameter File for Execution (see page 390)
- Batch Command Facility (z/VSE) (see page 391)
- Utility Statements (z/VSE) (see page 392)
- Utility Programs (z/VSE) (see page 416)
- IDMSLBLS Procedure for z/VSE installation (see page 423)

=COPY Facility

The =COPY IDMS statement is used to copy the library member into the job stream. Under z/VSE, some or all of the utility statements to be submitted to an IDMS utility can be stored as a member in a source statement library. Most IDMS utilities support the =COPY DIMS statement. This functionality is not available in CA IDMS Tools, CA Culprit, CA ADS batch, or CA OLQ batch.

The =COPY IDMS statement identifies the library member and is coded in the JCL along with other input parameter statements to be submitted to IDMS. Multiple =COPY statements can be submitted.

=COPY IDMS statements and input parameter statements can be intermixed in the JCL. The input parameters are submitted to the compiler in the order in which they occur, whether they are coded directly in the JCL or copied in through the =COPY facility.

Syntax

```
=COPY IDMS
  sublibrary-id.    member-name
  MEM= member-name
```

Parameters

- **sublibrary-id**
  Identifies the source statement sublibrary that includes the member identified by `member-name`. The default is A.
- **member-name**
  Identifies the source statement library member that contains the input parameter statements to be submitted to the IDMS utility.

- **member-type**
  Identifies the type of the member that contains the input parameter statements to be submitted to the IDMS utility. The default is A.

⚠️ **Note:** If the input parameter statements are stored as a member in a private source statement library, the DLBL file type for the library must be specified as DA.

---

**About the IDMSLBLs Procedure**

IDMSLBS is a procedure provided during a CA IDMS z/VSE installation. It contains file assignments for CA IDMS dictionaries, sample databases, disk journal files, and the SYSIDMS parameter file.

A copy of the IDMSLBS procedure (see page 423) appears at the end of this section.

File assignments for the CA IDMS files in the IDMSLBS procedure are not included in the sample JCL for each utility in this section.

Tape and archive journal files, SYSCTL, user database, and work file assignments are included in the sample JCL, as is an EXEC statement for the IDMSLBS procedure. The sample JCL assumes you will use either the IDMSLBS procedure or another procedure that you create.

The files required to run each utility are identified in each section under the JCL Considerations heading.

---

**SYSIDMS Parameter File for Execution**

SYSIDMS is a parameter file used in the execution of batch jobs running in either local mode or under the central version.

SYSIDMS parameters allow you to specify physical requirements of the environment, such as DBNAME and DICTNAME and runtime directives such as activating IDMSQSAM. Additionally, there are SYSIDMS parameters for use specifically in a z/VSE environment. For example, there is a BLKSIZE parameter that allows you to override the blocksize for a file and a DEVADDR=SYSnnn parameter to specify a device address for a tape file.

The SYSIDMS file can be defined as a sequential disk file, or SYSIDMS parameters can be passed through the SYSIPT file. The SYSIDMS file is defined in the IDMSLBS procedure so that parameters can be passed through SYSIPT.

SYSIDMS is referenced only in the generic JCL for the Batch Command Facility. You should include SYSIDMS parameters, as appropriate, in the JCL stream for each utility.
Note: On VSE systems, the default block size of the intermediate work files used by the CA IDMS utilities is 6000 bytes. If an intermediate work file contains records larger than 6000 bytes, you must use SYSIDMS to override the blocksize.

For example, in the UNLOAD utility, the record size of the SYS002 file is 24 bytes plus the size of the data portion of the record occurrence. If the total is greater than 6000 bytes, you can use the following SYSIDMS parameter as an override:

FILENAME=SYS002, BLKSIZE=32767, FILETYPE=D

Batch Command Facility (z/VSE)

The following sample z/VSE JCL is used to execute the Batch Command Facility (IDMSBCF).

When using the IDMSBCF program to execute a utility statement, include these file assignments along with the required statements for each of the utilities. The file assignments for each utility are presented on subsequent pages in this section.

Local mode IDMSBCF (z/VSE)

// LIBDEF *
// EXEC PROC=IDMSLBLS
// DLBL idmslib,'idms.library'
// EXTENT SYSnnn,nnnnnn,,ssss,tttt
// ASSGN SYSnnn,DISK,,VOL=vv,vv,vv,SHR
// TLBL SYSnnn,’idms.sysjrnl’,nnnnnn,,f
// ASSGN SYSnnn,x’cuu’

Insert file assignments required by the utility statements

// EXEC IDMSBCF,SIZE=64K

Insert SYSIDMS parameters, as required

/*

Insert utility statements

*/

Note: Additional file assignments may be needed for the user catalog and the system dictionary depending on your security implementation.

<table>
<thead>
<tr>
<th>IDMSLBLS</th>
<th>Name of the procedure provided at installation containing the file definitions for CA IDMS dictionaries, databases, and SYSIDMS parameter file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>idmslib</td>
<td>Dtftname of the CA IDMS library</td>
</tr>
</tbody>
</table>

Data set name of CA IDMS libraries, as established during installation
CA IDMS libraries

The CA IDMS libraries, as established during installation

SYSIDMS

Filename of the SYSIDMS parameter file

Note: For a complete listing of the SYSIDMS parameter file, see the CA IDMS Common Facilities Section.

Central version

To run IDMSBCF under the central version, add a SYSCTL file and remove the tape journal file assignment.

```
// LIBDEF *,SEARCH=CA IDMS libraries
// EXEC PROC=IDMSLBLDS
// DLBL  idmslib,'idms.library'
// EXTENT SYSnnn,nnnnnn,,,ssss,tttt
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL  sysctl,'idms.sysctl',1999/365,SD
// EXTENT SYSnnn,nnnnnn,,,ssss,tttt
// ASSGN  SYSnnn,DISK,VOL=nnnnnn,SHR

Insert file assignments required by the utility statements

// EXEC IDMSBCF,SIZE=64K

Insert SYSIDMS parameters, as required

/*

Insert utility statements

*/
```

sysctl

Filename of the SYSCTL file

idms.sysctl

File-ID of the SYSCTL file

Utility Statements (z/VSE)

ARCHIVE JOURNAL

File assignments for the batch command facility (z/VSE)

```
// TLBL  SYSnnn,'idms.archive',,nnnnnn,,f
// ASSGN  SYSnnn,x'cuu'

Additional archive journal file assignments, as required
```
### ARCHIVE LOG

File assignments for the batch command facility (z/VSE)

```plaintext
// DLBL   dclg,'idms.system.dldclog',,DA
// EXTENT  SYSnnn,nnnnnn,,ssss,llll
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// TLBL  SYS002,'idms.archive',,nnnnnn,,f
// ASSGN  SYS002,x'cuu'
```

dclg  File-ID of the database log file

idms.archive  File-ID of the archive log file

### ARCHIVE TRACE

The CA IDMS installation media contains the following sample JCL (https://docops.ca.com/display/IDMS/System+Generation+Syntax+Diagram+Conventions#SystemGenerationSyntaxDiagramConventions-SampleJCL) (intended for demonstration purposes only):

- ATRCJCL -- Contains a skeleton of the JCL required to run the ARCHIVE TRACE utility.

### BACKUP

File assignments for the batch command facility (z/VSE)

```plaintext
// DLBL   userdb,'user.userdb',,DA
// EXTENT  SYSnnn,nnnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL   userload
```

Additional file assignments, as required

```plaintext
// TLBL  SYS001,'user.bkpfile',,nnnnnn,,f
// ASSGN  SYS001,x'cuu'
```

userdb  Filename of the user file

user.userdb  File-ID of the user file

user.bkpfile  File-ID of the tape backup file

### BUILD

File assignments for the batch command facility (z/VSE)

```plaintext
// DLBL   userdb,'user.userdb',,DA
// EXTENT  SYSnnn,nnnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
```
Sort files are only needed when performing a complete BUILD.

// DLBL SORTWK1, 'sort.work.file'
// EXTENT SYSn,nnnnn,ssss,llll
// ASSGN SYSnnn,DISK,VOL=vvvvv,SHR

Additional sort files, as required

userdb Filename of the user file
userdb File-ID of the user file
user.load File-ID of the input (SYS002) file; for sizing information see discussion of SYS003 in LOAD (see page 109).
user.build File-ID of the output (SYS003) file; for sizing information see BUILD (see page 43).

⚠️ Note: When running a complete BUILD, SYS002 and SYS003 must point to the same intermediate file. When running a stepped BUILD, SYS002 and SYS003 must point to different intermediate files.

**CLEANUP**

File assignments for the batch command facility (z/VSE)

// DLBL userdb,'user.userdb',,DA
// EXTENT SYSnnn,nnnnn
// ASSGN SYSnnn,DISK,VOL=vvvvv,SHR

Additional database file assignments, as required

userdb Filename of the user file
userdb File-ID of the user file

Central version

To execute CLEANUP SEGMENT under the central version:

- Remove the TLBL and ASSGN statements for the tape journal file from the previous JCL.
- Insert the following statement:

// DLBL SYSCTL, file-id,0,SD
// EXTENT SYSnnn, vvvvv,1,0,1,1
// ASSGN SYSnnn,DISK,VOL=vvvvv,SHR
file-id Data set name of the sysctl file
SYSnnn Logical unit of the volume for which the extent is effective
vvvvv Volume serial number

**CONVERT CATALOG**

**DD statements for the batch command facility (z/VSE)**

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>//DLBL</td>
<td><code>ddlcat, 'ucat.ddlcat', DA</code></td>
</tr>
<tr>
<td>//EXTENT</td>
<td><code>SYSnnn, nnnnn</code></td>
</tr>
<tr>
<td>//ASSIGN</td>
<td><code>SYSnnn, DISK, VOL=vvvvvv, SHR</code></td>
</tr>
<tr>
<td>//DLBL</td>
<td><code>ddlcax, 'ucat.ddlcax', DA</code></td>
</tr>
<tr>
<td>//EXTENT</td>
<td><code>SYSnnn, nnnnn</code></td>
</tr>
<tr>
<td>//ASSIGN</td>
<td><code>SYSnnn, DISK, VOL=vvvvvv, SHR</code></td>
</tr>
</tbody>
</table>

**Central version**

To execute CONVERT CATALOG under the central version:

- Optionally remove any journal and database file statements
- Insert the following statement:

  ```
  //DLBL SYSCTL, file-id, _, SD
  //EXTENT SYSnnn, nnnnnn
  //ASSIGN SYSnnn, DISK, VOL=vvvvvv, SHR
  ```

**CONVERT PAGE**

**File assignments for the batch command facility (z/VSE)**

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>//DLBL</td>
<td><code>userdb, 'user.userdb', SD</code></td>
</tr>
<tr>
<td>//EXTENT</td>
<td><code>SYSnnn, nnnnnn</code></td>
</tr>
<tr>
<td>//ASSIGN</td>
<td><code>SYSnnn, DISK, VOL=vvvvvv, SHR</code></td>
</tr>
</tbody>
</table>

Additional database file assignments as needed.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>//DLBL</td>
<td><code>newdb, 'user.newdb', SD</code></td>
</tr>
<tr>
<td>//EXTENT</td>
<td><code>SYSnnn, nnnnnn</code></td>
</tr>
</tbody>
</table>
//ASSIGN SYSnnn,DISK,VOL=vvvvv,SHR

Additional new converted database file assignments as needed.

<table>
<thead>
<tr>
<th>userdb</th>
<th>Filename of the input database file</th>
</tr>
</thead>
<tbody>
<tr>
<td>user.userdb</td>
<td>File-ID of the input database file</td>
</tr>
<tr>
<td>newdb</td>
<td>Filename of the output converted database file</td>
</tr>
<tr>
<td>user.newdb</td>
<td>File-ID of the output converted database file</td>
</tr>
</tbody>
</table>

EXPAND PAGE

File assignments for the batch command facility (z/VSE)

// DLBL userdb,'user.userdb',,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN SYSnnn,DISK,VOL=vvvvv,SHR

Additional database file assignments, as required

// DLBL userxb,'user.xbase',,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN SYSnnn,DISK,VOL=vvvvv,SHR

Additional expanded database file assignments, as required

<table>
<thead>
<tr>
<th>user.userdb</th>
<th>File-ID of the existing database file</th>
</tr>
</thead>
<tbody>
<tr>
<td>user.xbase</td>
<td>File-ID of the expanded database file</td>
</tr>
</tbody>
</table>

EXTRACT JOURNAL

File assignments for the batch command facility (z/VSE)

// DLBL SYS001,'archive',,SD
// EXTENT SYS001,nnnnnn
// ASSGN SYS001,DISK,VOL=vvvvv,SHR
// DLBL extract,'jrnl.extract',,SD
// EXTENT sysnnn,nnnnnn
// ASSGN sysnnn,DISK,VOL=vvvvv,SHR

Add sort work files

Add SYSIDMS parameters

FILENAME=SYSnnn, BLKSIZE=bbbb

archive File name of the complete archive or journal file. It can be on tape or disk as specified through the SYSIDMS parameter file.

extract DDname of the extract journal file. If not specified, it defaults to SYS002. It can be on tape or disk as specified through the SYSIDMS parameter file.

eaurl Data set name of the extract journal file.
Block size of the extract journal file. Specify a size as large as the largest block size on the journal or archive files being processed.

**FASTLOAD**

File assignments for the batch command facility (z/VSE)

```
// DLBL  SORTWK1,'sort.work.file'
// EXTENT SYSnnn,nnnnnn,,ssss,llll
// ASSGN  SYSnnn,DISK, VOL=vvvvv, SHR
```

Additional sort files, as required

```
// DLBL  userdb,'user.userdb'.,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN  SYSnnn,DISK, VOL=vvvvv, SHR
```

Additional database file assignments, as required

```
// DLBL  RELDCTL,'user.reldctl',,SD
// EXTENT SYSnnn,nnnnnn,,ssss,llll
// ASSGN  SYSnnn,DISK, VOL=vvvvv, SHR
// DLBL  SYS001,'sort.unload',,SD
// EXTENT SYS001,nnnnnn,,ssss,llll
// ASSGN  SYS001,DISK, VOL=vvvvv, SHR
// DLBL  SYS002,'user.db1001',,SD
// EXTENT SYS002,nnnnnn,,ssss,llll
// ASSGN  SYS002,DISK, VOL=vvvvv, SHR
// DLBL  SYS004,'db1004.name',,SD
// EXTENT SYS004,nnnnnn,,ssss,llll
// ASSGN  SYS004,DISK, VOL=vvvvv, SHR
// DLBL  SYS005,'user.db1005',,SD
// EXTENT SYS005,nnnnnn,,ssss,llll
// ASSGN  SYS005,DISK, VOL=vvvvv, SHR
// TLBL  SYS009,'user.db1009',,nnnnnn,,f
// ASSGN  SYS009,'x'cuu'
// DLBL  SYS010,'user.db1010',,SD
// EXTENT SYS010,nnnnnn,,ssss,llll
// ASSGN  SYS010,DISK, VOL=vvvvv, SHR
// DLBL  SYS011,'user.db1011',,SD
// EXTENT SYS011,nnnnnn,,ssss,llll
// ASSGN  SYS011,DISK, VOL=vvvvv, SHR
```

**sor** File-ID of the sort parameters created by IDMSTBLU

**t.**

**unl**

**oa**

**d**

**use** Data set name of the reload control file containing control and set information. Blocksize of the **reldctl** file should be a multiple of 60 with a maximum size of 32,760. Blocksize is controlled through the BLKSIZE SYSIDMS parameter. For a complete description of SYSIDMS parameters, see the **CA IDMS Common Facilities Section**.

**dct**

**t.**

**use** File-ID of the file generated by the format program

**r.**

**dbl**

**00**

**1**
use File-ID of the intermediate work file containing the output from SORT1. Record and block size should be the same as SYS001.

dbl
00
4

use File-ID of the intermediate work file containing a control record and set membership information from IDMSDBL2.

dbl
00
5

use File-ID of the intermediate work file containing the sorted contents of SYS005.

r. 
dbl
00
9

use File-ID of the intermediate work file containing pointer information from IDMSDBL3.

r. 
dbl
01
0

use File-ID of the intermediate work file containing sorted pointer information from SORT4.

r. 
dbl
01
1

use Filename of the database file.

rdb

use File-ID of the database file.

r.
use 
rdb

---

**FIX ARCHIVE**

File assignments for the batch command facility (z/VSE)

```
// TLBL    SYS001,'idms.tjrnlold',,nnnnnn,,f
// ASSGN   SYS001, 'x'cuu'
// TLBL    SYS002,'idms.tjrnlfix',,nnnnnn,,f
// ASSGN   SYS002, 'x'cuu'
```

<table>
<thead>
<tr>
<th>File-ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>idms.tjrnlold</td>
<td>File-ID of the input tape journal file</td>
</tr>
<tr>
<td>f</td>
<td>File number of the tape journal file</td>
</tr>
<tr>
<td>idms.tjrnlfix</td>
<td>File-ID of the output tape journal file</td>
</tr>
</tbody>
</table>
FIX PAGE

File assignments for the batch command facility (z/VSE)

// DLBL userdb,'user.userdb',,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR

Additional file assignments, as required

<table>
<thead>
<tr>
<th>userdb</th>
<th>Filename of the user file</th>
</tr>
</thead>
<tbody>
<tr>
<td>user.userdb</td>
<td>File-ID of the user file</td>
</tr>
</tbody>
</table>

Central version

To execute FIX PAGE under the central version:

- Remove the TLBL and ASSGN statements for the tape journal file from the previous JCL.
- Insert the following statement:

```plaintext
// DLBL SYSCTL, file-id,0,SD
// EXTENT SYSnnn,vvvvvv,1,0,1,1
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

<table>
<thead>
<tr>
<th>file-id</th>
<th>Data set name of the sysctl file</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSnnn</td>
<td>Logical unit of the volume for which the extent is effective</td>
</tr>
<tr>
<td>vvvvvv</td>
<td>Volume serial number</td>
</tr>
</tbody>
</table>

FORMAT

File assignments for the batch command facility (z/VSE)

To format a new database file:

```plaintext
// DLBL userdb,'user.userdb',,tt
// EXTENT SYSnnn,nnnnnn,,ssss,rrrr
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

To reformat an existing database file:

```plaintext
// DLBL userdb,'user.userdb',,tt
// EXTENT SYSnnn,nnnnnn
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

To format a new disk journal file:

```plaintext
// DLBL j1jrnl,'idms.j1jrnl',,SD
// EXTENT SYSnnn,nnnnnn,,ssss,rrrr
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

To reformat an existing disk journal file:
Additional database and journal file assignments, as required

To format a new SYSTRK file:

// DLBL  ffff.,'idms.systrk1',,SD
// EXTENT  SYSnnn,nnnnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR

To reformat an existing SYSTRK file:

// DLBL  ffff.,'idms.systrk1',,xx
// EXTENT  SYSnnn,nnnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR

userdb  Filename of the database file
userdb  File-ID of the database file
userdb  File-ID of the disk journal file
j1jrnl  Filename of the disk journal file
rrrr  Number of tracks (CKD) or blocks (FBA) in the disk extent; must be at least one more than the number of BDAM blocks or VSAM control intervals in the database
tt  File type of the file being formatted:
SD (sequential) if formatting by file or segment (unless it is a VSAM file)
DA (direct access) if reformattting by area
VSAM if formatting or reformatting a CA IDMS/DB VSAM file
Omit if reformatting by file
ffff  File name of the SYSTRK file
idms.  FILE-ID of the SYSTRK file
systrk1
xx  File type of the file being formatted
SD (sequential) if INITIAL is specified
DA (direct access) if INITIAL is not specified
VSAM if formatting a VSAM file

Multiple extents

You can format a CA IDMS/DB VSAM file with multiple extents under z/VSE. However, you cannot format a BDAM file with multiple extents.

The SIZE parameter on the EXEC statement

The SIZE parameter on the EXEC statement enables the IBM rotational sensing (RPS) feature. When formatting CA IDMS/DB VSAM files, you should specify SIZE=AUTO on the EXEC statement for IDMSBCF.
Do not use the SIZE parameter at all when formatting BDAM files or files allocated on 3344 disk devices with RPS implemented within the hardware. When you use the SIZE parameter in these cases, CA IDMS/DB cannot access the required current updated copy of the DTF.

**Central version**

To execute FORMAT AREA or FORMAT SEGMENT under the central version:

- Remove the TLBL and ASSGN statements for the tape journal file from the previous JCL.

- Insert the following statement:

```plaintext
// DLBL SYSCTL,file-id,0,SD
// EXTENT SYSnnn,vvvvvv,1,0,1,1
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

<table>
<thead>
<tr>
<th><strong>file-id</strong></th>
<th>Data set name of the sysctl file</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SYSnnn</strong></td>
<td>Logical unit of the volume for which the extent is effective</td>
</tr>
<tr>
<td><strong>vvvvvv</strong></td>
<td>Volume serial number</td>
</tr>
</tbody>
</table>

**INSTALL STAMPS**

**Local mode**

File assignments for the batch command facility (z/VSE)

```plaintext
// DLBL userdb,'user.userdb',,tt
// EXTENT SYSnnn,nnnnnn,,ssss,rrrr
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL userdict,'user.userdict',,tt
// EXTENT SYSnnn,nnnnnn
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

<table>
<thead>
<tr>
<th><strong>userdb</strong></th>
<th>Filename of the user file</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>user.userdb</strong></td>
<td>File-ID of the user file</td>
</tr>
<tr>
<td><strong>user.userdict</strong></td>
<td>File-ID of the dictionary containing the table definitions</td>
</tr>
</tbody>
</table>

**Central version**

To execute INSTALL STAMPS under the central version:

- Remove the TLBL and ASSGN statements for the tape journal file from the previous JCL.

- Insert the following statement:

```plaintext
// DLBL SYSCTL,file-id,0,SD
// EXTENT SYSnnn,vvvvvv,1,0,1,1
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

| **file-id** | Data set name of the sysctl file |
SYSnnn  Logical unit of the volume for which the extent is effective
vvvvv  Volume serial number

LOAD

File assignments for the batch command facility (z/VSE)

// DLBL  userdb,'user.userdb',,DA
// EXTENT  SYSnnn,nmnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL  userdict,'user.userdict',,tt
// EXTENT  SYSnnn,nmnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL  SYS001,'user.input',,SD
// EXTENT  SYS001,nmnmnnn,,sss,,lllll
// ASSGN  SYS001,DISK,VOL=vvvvvv,SHR
// DLBL  SYS002,'user.loadin',,SD
// EXTENT  SYS002,nfnmnnn,,sss,,llll
// ASSGN  SYS002,DISK,VOL=vvvvvv,SHR
// DLBL  SYS003,'user.loadout',,SD
// EXTENT  SYS003,nfnmnnn,,sss,,llll
// ASSGN  SYS003,DISK,VOL=vvvvvv,SHR
// DLBL  SYS004,'sort.load',,SD
// EXTENT  SYS004,nfnmnnn,,sss,,llll
// ASSGN  SYS004,DISK,VOL=vvvvvv,SHR

Sort files are only needed when performing a complete LOAD

// DLBL  SYSnnn,'sort.work',,SD
// EXTENT  SYSnnn,nmnmnnn,,sss,,llll
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR

Additional sort files, as required

userdb  Filename of the user file
user.userdb  File-ID of the user file
user.userdict  File-ID of the dictionary containing the table definitions
user.input  File-ID of the input file
user.loadin  File-ID of the input SYS002 file; for sizing information see LOAD (see page 109).
user.loadout  File-ID of the output SYS003 file; for sizing information see LOAD (see page 109).
sort.load  File-ID of the SYSPCH file

⚠️ Note: When performing a complete LOAD, you must preallocate the file referenced by SYS002 and SYS003 (the same file is used for both assigns in a complete LOAD). Do not use a temporary data set for these files.
LOCK

File assignments for the batch command facility (z/VSE)

// DLBL userdb,'user.userdb',,DA
// EXTENT SYSnnp,nnnnnn
// ASSGN SYSnnp,DISK,VOL=vvvvvv,SHR

user.userdb  File-ID of the user database file

Central version

To execute LOCK AREA under the central version:

- Remove the TLBL and ASSGN statements for the tape journal file from the previous JCL.

- Insert the following statement:

  // DLBL SYSCtl,file-id,0,SD
  // EXTENT SYSnnp,vvvvvv,1,0,1,1
  // ASSGN SYSnnp,DISK,VOL=vvvvvv,SHR

file-id  Data set name of the syctl file
SYSnnp  Logical unit of the volume for which the extent is effective
vvvvvv  Volume serial number

MAINTAIN INDEX

File assignments for the batch command facility (z/VSE)

// DLBL SORTWK1,'sort.work.file'
// EXTENT SYSnnp,nnnnnn,,ssss,llll
// ASSGN SYSnnp,DISK,VOL=vvvvvv,SHR

Additional sort files, as required

// DLBL userdb,'user.olddb',,DA
// EXTENT SYSnnp,nnnnnn
// ASSGN SYSnnp,DISK,VOL=vvvvvv,SHR

Additional existing database file assignments, as required

// DLBL RELDCTL.'user.reldctl',,SD
// EXTENT SYSnnp,nnnnnn,,ssss,llll
// ASSGN SYSnnp,DISK,VOL=vvvvvv,SHR
// DLBL SYS003,'user.db1003',,SD
// EXTENT SYS003,nnnnnn,,ssss,llll
// ASSGN SYS003,DISK,VOL=vvvvvv,SHR
// DLBL SYS004,'user.db1004',,SD
// EXTENT SYS004,nnnnnn,,ssss,llll
// ASSGN SYS004,DISK,VOL=vvvvvv,SHR
// DLBL SYS005,'user.db1005',,SD
// EXTENT SYS005,nnnnnn,,ssss,llll
// ASSGN SYS005,DISK,VOL=vvvvvv,SHR
// DLBL SYS006,'user.db1006',,SD
// EXTENT  SYS006,nnnnnn,ssss,llll
// ASSGN   SYS006,DISK,VOL=vvvvvv,SHR

**us**  File-ID of the existing database file  
**er.**
**old**
**db**

**us**  Data set name of the reload control file containing control and set information. Block size of the  
**er.**  RELDCTL file should be a multiple of 60 with a maximum size of 32,760. Block size is controlled  
**rel**  through the BLKSIZE SYSIDMS parameter. For a complete description of SYSIDMS parameters, see  
**dct**  the *CA IDMS Common Facilities Section*.

**us**  File-ID of the intermediate work file containing index descriptors from IDMSTABX.  
**er.**
**dbl**
**00**
**3**

**us**  File-ID of the intermediate work file containing the output from SORT3.  
**er.**
**dbl**
**00**
**4**

**us**  File-ID of the intermediate work file containing pointers for user-owned index sets from  
**er.**  IDMSDBL3.  
**dbl**
**00**
**5**

**us**  File-ID of the intermediate work file containing sorted pointers for user-owned index sets from  
**er.**  SORT4.  
**dbl**
**00**
**6**

---

**PRINT INDEX**

**File assignments for the batch command facility (z/VSE)**

// DLBL   userdb,'user.userdb','DA
// EXTENT SYSnnn,nnnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR

*Additional file assignments, as required*

**userdb**  Filename of the user file

**user.userdb**  File-ID of the user file
PRINT JOURNAL

File assignments for the batch command facility (z/VSE)

// DLBL     SYS001,'idms.archjrnl',SD
// EXTENT   SYSnnn,nnnnnn
// ASSGN    SYSnnn,DISK,VOL=vvvvvv,SHR

SYS001              Filename of the archive journal file
idms.archjrnl      File-ID of the archive journal file

PRINT LOG

File assignments for the batch command facility (z/VSE)

To print from the DDLDCLOG area:

// DLBL     dclog,'idms.system.ddldclog',,DA
// EXTENT   SYSnnn,nnnnnn,,ssss,llll
// ASSGN    SYSnnn,DISK,VOL=vvvvvv,SHR

To print from the archive log file:

// TLBL     SYS001,'idms.archive',,nnnnnn,,f
// ASSGN    SYS001,x'cuu'

idms.archive      File-ID of the archive log file

PRINT PAGE

File assignments for the batch command facility (z/VSE)

// DLBL     userdb1,'user.userdb1',,DA
// EXTENT   SYSnnn,nnnnnn
// ASSGN    SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL     userdb2,'user.userdb2',,DA
// EXTENT   SYSnnn,nnnnnn
// ASSGN    SYSnnn,DISK,VOL=vvvvvv,SHR

Additional file assignments, as required

userdb1             Filename of the user file
user.userdb1        File-ID of the user file

Central version

To execute PRINT PAGE under the central version:

- Remove the TLBL and ASSGN statements for the tape journal file from the previous JCL.
Insert the following statement:

```
// DLBL SYSCTL, file-id, 0, SD
// EXTENT SYSnnn, vvvvvv, 1,0,1,1
// ASSGN SYSnnn, DISK, VOL=vvvvvv, SHR
```

<table>
<thead>
<tr>
<th>file-id</th>
<th>Data set name of the sysctl file</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSnnn</td>
<td>Logical unit of the volume for which the extent is effective</td>
</tr>
<tr>
<td>vvvvvv</td>
<td>Volume serial number</td>
</tr>
</tbody>
</table>

**PRINT SPACE**

File assignments for the batch command facility (z/VSE)

```
// DLBL userdb1, 'user.userdb1', , DA
// EXTENT SYSnnn, nnnnnn
// ASSGN SYSnnn, DISK, VOL=vvvvvv, SHR
// DLBL userdb2, 'user.userdb2', , DA
// EXTENT SYSnnn, nnnnnn
// ASSGN SYSnnn, DISK, VOL=vvvvvv, SHR
```

Additional file assignments, as required

<table>
<thead>
<tr>
<th>userdb1</th>
<th>Filename of the user file</th>
</tr>
</thead>
<tbody>
<tr>
<td>user.userdb1</td>
<td>File-ID of the user file</td>
</tr>
</tbody>
</table>

**Central version**

To execute PRINT SPACE FOR AREA or PRINT SPACE FOR SEGMENT under the central version:

- Remove the TLBL and ASSGN statements for the tape journal file from the previous JCL.
- Insert the following statement:

```
// DLBL SYSCTL, file-id, 0, SD
// EXTENT SYSnnn, vvvvvv, 1,0,1,1
// ASSGN SYSnnn, DISK, VOL=vvvvvv, SHR
```

<table>
<thead>
<tr>
<th>file-id</th>
<th>Data set name of the sysctl file</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSnnn</td>
<td>Logical unit of the volume for which the extent is effective</td>
</tr>
<tr>
<td>vvvvvv</td>
<td>Volume serial number</td>
</tr>
</tbody>
</table>

**PRINT TRACE**

The CA IDMS installation media contains the following sample JCL:

```
(PTRCJCL - Contains a skeleton of the JCL required to run the PRINT TRACE utility)
```

(intended for demonstration purposes only):
Local mode

File assignments for the batch command facility (z/VSE)

// DLBL  usercat,'user.ddlcat',,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL  usercatl,'user.ddlcatl',,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL  usercatx,'user.ddlcatx',,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// TLBL  SYSnnn,'idms.sysjrnl',,nnnnnn,,f
// ASSGN  SYSnnn,x’cuu’

Additional dummy journals, as required

user.ddlcat  File-ID of the dictionary file containing the DDLCAT area of the dictionary
user.ddlcatl File-ID of the dictionary file containing the DDLCATLOD area of the dictionary
user.ddlcatx File-ID of the dictionary file containing the DDLCATX area of the dictionary
idms.sysjrnl File-ID of the user file

Central version

To execute INSTALL STAMPS under the central version, remove the USERCAT, USERCATL, USERCATX, and SYSJRNL statements.

RELOAD

File assignments for the batch command facility (z/VSE)

// DLBL  SORTWK1,'sort.work.file'
// EXTENT SYSnnn,nnnnnn,,ssss,llll
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR

Additional sort files, as required

// DLBL  userdb,'.user.userdb',,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR

Additional existing database file assignments, as required

// DLBL  RELDCTL.'user.reldctl',,SD
// EXTENT SYSnnn,nnnnnn,,ssss,llll
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL  SYS001,'sort.unload',,SD
// EXTENT SYS001,nnnnnn,,ssss,llll
// ASSGN  SYS001,DISK,VOL=vvvvvv,SHR
// DLBL  SYS002,'user.db1002',,SD
// EXTENT SYS002,nnnnnn,,ssss,llll
// ASSGN  SYS002,DISK,VOL=vvvvvv,SHR
// DLBL  SYS003,'user.db1003',,SD
// EXTENT SYS003,nnnnnn,,ssss,llll
// ASSGN  SYS003,DISK,VOL=vvvvvv,SHR
// DLBL  SYS004,'user.db1004',,SD
// EXTENT SYS004,nnnnnn,,ssss,llll
// ASSGN  SYS004,DISK,VOL=vvvvvv,SHR
sor File-ID of the sort parameters created by UNLOAD.

t. unl

oa d

data Data set name of the reload control file containing control and set information. Blocksize of the
r. RELDCTL file should be a multiple of 60 with a maximum size of 32,760. Blocksize is controlled
rel through the BLKSIZE SYSIDMS parameter. For a complete description of SYSIDMS parameters, see
dct the CA IDMS Common Facilities Section.

use File-ID of the file generated by UNLOAD as SYS001.

r. dbl

00 2

use File-ID of the file generated by UNLOAD as SYS003.

r. dbl

00 3

use File-ID of the intermediate work file containing the output from SORT1.

r. dbl

00 4

use File-ID of the intermediate work file containing member descriptors for chained sets from
r. IDMSDBL2.

00 5

File-ID of the intermediate work file containing member descriptors for user-owned index sets
from IDMSDBL2.
**REORG**

**DD statements for the batch command facility (z/VSE) Defining the REORG control file, all jobs and job steps**

```plaintext
// DLBL RORGCTL,’user.rorgctl’,,SD
// EXTENT SYSnn,vvvvv,,sssss,llll
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
```

**Defining the REORG JCL file, required when submitting jobs**
Manual definition of work files, when not using DSMODELs

// DLBL RORGJCL,'user.jclfile',,SD
// EXTENT SYSnnn,vvvvv,,,sssss,llll
// ASSGN SYSnnn,DISK,VOL=vvvvv,SHR

Additional work file definitions, as required

DB File definitions when running the unload phase, if not using dynamic allocation

// DLBL wxnnnnn,'user.workfile',,SD llll
// EXTENT SYSnnn,svvsvv,ssssss
// ASSGN SYSnnn,DISK,VOL=vvvvv,SHR

Additional unload database file definitions, as required

DB File definitions when running the reload phase, if not using dynamic allocation

// DLBL relldb,'user.relddb',,DA
// EXTENT SYSnnn,svvsvv
// ASSGN SYSnnn,DISK,VOL=vvvvv,SHR

Additional reload database file definitions, as required

SORT work file assignments when running the reload phase

// DLBL SORTWK1,'sort.work.file'
// EXTENT SYSnnn,svvsvv,ssssss,llll
// ASSGN SYSnnn,DISK,VOL=vvvvv,SHR

Additional sort work file definitions, as required

- **user.reldctl**
  File-ID of the REORG control file containing control information. The block size is 8192 bytes, and must be specified in SYSIDMS using the FILENAME=RORGCTL RECFM=F BLKSIZE=8192 parameters.

  - **Note**: For more information about SYSIDMS parameters, see the CA IDMS Common Facilities Section.

- **user.jclfile**
  File-ID of the file containing JCL for automatic job submission. The block size must be a multiple of 80 bytes, and must be specified in SYSIDMS using the FILENAME=RORGJCL RECFM=F BLKSIZE= parameters.

- **wxnnnnn**
  File name of the DLBL for a work file. It must match the name generated in the Unload/Reload Work File Summary report.

- **user.workfile**
  File-ID of a work file when sectionly allocating work files.
- **unlddb**
  File name of the DLBL for an unload database file.

- **user.unlddb**
  File-ID of the unload database file, this is source database file.

- **relddb**
  File name of the DLBL for a reload database file.

- **user.relddb**
  File-ID of the reload database file, this is the target database file.

### RESTORE

**File assignments for the batch command facility (z/VSE)**

```
// DLBL  userdb1,'user.userdb1',,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL  userdb2,'user.userdb2',,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// TLBL  SYS001,'user.bkpfile'
// ASSGN SYS001,x'cuu'
```

*Additional file assignments, as required*

<table>
<thead>
<tr>
<th><strong>userdb1</strong></th>
<th>Filename of the user file</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>user.userdb1</strong></td>
<td>File-ID of the user file</td>
</tr>
<tr>
<td><strong>user.bkpfile</strong></td>
<td>File-ID of the backup file</td>
</tr>
</tbody>
</table>

### RESTRUCTURE

**File assignments for the batch command facility (z/VSE)**

```
// DLBL  userdb1,'user.userdb1',,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL  userdb2,'user.userdb2',,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL  SYS001,'user.spill',,DA
// EXTENT SYS001,nnnnnn
// ASSGN SYS001,DISK,VOL=vvvvvv,SHR
```

*Additional file assignments, as required*

<table>
<thead>
<tr>
<th><strong>userdb1</strong></th>
<th>Filename of the user file</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>user.userdb1</strong></td>
<td>File-ID of the user file</td>
</tr>
<tr>
<td><strong>user.spill</strong></td>
<td>File-ID of the spill file; for sizing information see <a href="#">RESTRUCTURE CONNECT</a> (see page 248)</td>
</tr>
</tbody>
</table>
RESTRUCTURE CONNECT

File assignments for the batch command facility (z/VSE)

// DLBL     userdb,'user.userdb',,DA
// EXTENT   SYSnnn,nnnnnn
// ASSGN    SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL     SYS001,'user.spill',,DA
// EXTENT   SYS001,nnnnnn
// ASSGN    SYS001,DISK,VOL=vvvvvv,SHR
// DLBL     SORTWK1,'sort.work.file'
// EXTENT   SYSnnn,nnnnnn,ssss,llll
// ASSGN    SYSnnn,DISK,VOL=vvvvvv,SHR

Additional sort files, as required

user.userdb  File-ID of the user database file
user.spill   File-ID of the spill file; for sizing information see RESTRUCTURE CONNECT (see page 248).

ROLLBACK

File assignments for the batch command facility (z/VSE)

// DLBL     userdb,'user.userdb',,DA
// EXTENT   SYSnnn,nnnnnn
// ASSGN    SYSnnn,DISK,VOL=vvvvvv,SHR

Additional database file assignments, as required

// TLBL     SYS001,'archive'
// ASSGN    SYS001,x'cuu'

If using the SORT option, add sort files

userdb      Filename of the user file
archive     File-ID of the complete archive or tape journal file

ROLLFORWARD

File assignments for the batch command facility (z/VSE)

// DLBL     userdb,'user.userdb',,DA
// EXTENT   SYSnnn,nnnnnn
// ASSGN    SYSnnn,DISK,VOL=vvvvvv,SHR

Add database file assignments, as required

If recovering from a standard journal file

// TLBL     SYSnnn,'archive'
// ASSGN    SYSnnn,X'cuu'

If recovering from a journal extract file
If recovering from a journal extract file

// DLBL extract,'jrnl.extract',SD  
// EXTENT SYSnnn,nnnnnn  
// ASSGN SYSnnn,DISK,VOL=vvvvv,SHR

If using the SORT option or processing a journal extract file, add sort files as required

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userdb</td>
<td>Filename of the user file.</td>
</tr>
<tr>
<td>archive</td>
<td>File-ID of the complete archive or tape journal file.</td>
</tr>
<tr>
<td>extract</td>
<td>File name of the extract journal file. If you do not specify, it defaults to SYS002.</td>
</tr>
<tr>
<td>jrnl.extract</td>
<td>File-ID of the extract journal file.</td>
</tr>
</tbody>
</table>

**SYNCHRONIZE STAMPS**

**Local mode**

File assignments for the batch command facility (z/VSE)

// DLBL userdb,'user.userdb',tt  
// EXTENT SYSnnn,nnnnnn,ssss,rrrr  
// ASSGN SYSnnn,DISK,VOL=vvvvv,SHR  
// DLBL userdict,'user.userdict',tt  
// EXTENT SYSnnn,nnnnnn  
// ASSGN SYSnnn,DISK,VOL=vvvvv,SHR

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userdb</td>
<td>Filename of the user file.</td>
</tr>
<tr>
<td>user.userdb</td>
<td>File-ID of the user file.</td>
</tr>
<tr>
<td>user.userdict</td>
<td>File-ID of the dictionary containing the table definitions</td>
</tr>
</tbody>
</table>

**Central version**

To execute SYNCHRONIZE STAMPS under the central version:

- Remove the TLBL and ASSGN statements for the tape journal file from the previous JCL.
- Insert the following statement:

// DLBL SYSCTL,file-id,0,SD  
// EXTENT SYSnnn.vvvvvv,1,0,1,1  
// ASSGN SYSnnn,DISK,VOL=vvvvv,SHR

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file-id</td>
<td>Data set name of the sysctl file</td>
</tr>
<tr>
<td>SYSnnn</td>
<td>Logical unit of the volume for which the extent is effective</td>
</tr>
<tr>
<td>vvvvvv</td>
<td>Volume serial number</td>
</tr>
</tbody>
</table>
TUNE INDEX

File assignments for the batch command facility (z/VSE)

//DLBL  userdb,'user.userdb',,DA
//EXTENT SYSnnn,nnnnnn
//ASSIGN SYSnnn,DISK,VOL=vvvvvv,SHR

<table>
<thead>
<tr>
<th>userdb</th>
<th>Filename of the user file</th>
</tr>
</thead>
<tbody>
<tr>
<td>user.userdb</td>
<td>File-ID of the user file</td>
</tr>
</tbody>
</table>

Central version

To execute TUNE INDEX under the central version:

- Remove the TLBL and ASSGN statements for the tape journal file from the previous JCL.

- Insert the following statement:
  
  // DLBL SYSCTL,file-id,0,SD
  // EXTENT SYSnnn,vvvvvv,1,0,1,1
  // ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR

<table>
<thead>
<tr>
<th>file-id</th>
<th>Data set name of the sysctl file</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSnnn</td>
<td>Logical unit of the volume for which the extent is effective</td>
</tr>
<tr>
<td>vvvvvv</td>
<td>Volume serial number</td>
</tr>
</tbody>
</table>

UNLOCK

File assignments for the batch command facility (z/VSE)

// DLBL  userdb,'user.userdb',,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR

| user.userdb | File-ID of the user database file |

UNLOAD

File assignments for the batch command facility (z/VSE)

// DLBL  SORTWK1,'sort.work.file'
// EXTENT SYSn,nnnnnn,,ssss,lli
// ASSGN SYSn,DISK,VOL=vvvvvv,SHR

<table>
<thead>
<tr>
<th>Additional sort files, as required</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>// DLBL  RELDCTL.'user.reldctl',,SD</td>
<td></td>
</tr>
</tbody>
</table>
File assignments for the batch command facility (z/VSE)

```
// DLBL   userdb,'user.userdb',,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// TLBL   SYSnnn,'idms.tapejrnl',,nnnnnn,f
// ASSGN  SYSnnn,x'cuu'
```

```
userdb        Filename of the user file
```

Central version

To execute UPDATE STATISTICS under the central version:

- Remove the TLBL and ASSGN statements for the tape journal file from the previous JCL.
- Insert the following statement:
// DLBL SYSCTL,file-id,0,SD
// EXTENT SYSnnn,vvvvvv,1,0,1,1
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR

<table>
<thead>
<tr>
<th>file-id</th>
<th>Data set name of the sysctl file</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSnnn</td>
<td>Logical unit of the volume for which the extent is effective</td>
</tr>
<tr>
<td>vvvvvv</td>
<td>Volume serial number</td>
</tr>
</tbody>
</table>

**VALIDATE**

**File assignments for the batch command facility (z/VSE)**

// DLBL userdb,'user.userdb',,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL SYS002,'user.valin',,DA
// EXTENT SYS002,nnnnnn
// ASSGN SYS002,DISK,VOL=vvvvvv,SHR
// DLBL SYS003,'user.valout',,DA
// EXTENT SYS003,nnnnnn
// ASSGN SYS003,DISK,VOL=vvvvvv,SHR
// DLBL SYSnnn,'sort.load',,SD
// EXTENT SYSnnn,nnnnnn,,ssss,llll
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR

If performing a complete VALIDATE, add sort files as necessary

// DLBL SYSnnn,'sort.work',,SD
// EXTENT SYSnnn,nnnnnn,,ssss,llll
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR

**user.userdb** File-ID of the database file

**user.valin** File-ID of the input SYS002 file; for sizing information see VALIDATE (see page 295).

**user.valout** File-ID of the SYS003 output file; for sizing information see VALIDATE (see page 295).

**sort.load** File-ID of the SYSPCH file

⚠️ **Note:** When running a complete VALIDATE, SYS002 and SYS003 must point to the *same* intermediate file. When running a stepped VALIDATE, SYS002 and SYS003 must point to a *different* intermediate file.

**Utility Programs (z/VSE)**

This topic lists the following utility programs:

- IDMSDBAN (see page 417)
- IDMSDIRL (see page 418)
- IDMSLOOK (see page 419)
IDMSDBAN

IDMSDBAN (z/VSE)
// DLBL  idmslib,'idms.library'
// EXTENT SYSnnn,nnnnnn,ssss,tttt
// ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR
// LIBDEF *,SEARCH=CA IDMS libraries
// EXEC PROC=IDMSLBLS
// DLBL  userdb,'user.userdb',,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR

Additional database file assignments, as required
// TLBL  SYS002,'chain.file',,nnnnnn,f
// ASSGN SYS002,x'cuu'
// EXEC  IDMSDBN1

Insert SYSIDMS parameters, as required
/

IDMSDBAN input parameters
// ASSGN SYS004,x'cuu'
// EXTENT SYS004

Additional sort work file assignments, as required
// TLBL  SYS001,'chain.file',,nnnnnn,f
// ASSGN SYS001,x'cuu'
// TLBL  SYSnnn,'output.work',,nnnnnn,f
// ASSGN SYSnnn,x'cuu'
// TLBL  SYSnnn,'input.work',,nnnnnn,f
// ASSGN SYSnnn,SYS002
// EXEC  IDMSDBN2

Insert SYSIDMS parameters, as required
/

IDMSLBLS  Name of the procedure provided at installation containing the file definitions for CA IDMS dictionaries, databases, and SYSIDMS parameter file.

Note: For a complete listing of IDMSLBLS, see "IDMSLBLS Procedure" later in this section.

<table>
<thead>
<tr>
<th>idmslib</th>
<th>Dftname of the CA IDMS library</th>
</tr>
</thead>
<tbody>
<tr>
<td>idms.library</td>
<td>Data set name of CA IDMS libraries, as established during installation</td>
</tr>
<tr>
<td>SYSnnn</td>
<td>Logical unit of the volume for which the extent is effective</td>
</tr>
<tr>
<td>nnnnnn</td>
<td>Volume serial identifier of appropriate disk volume</td>
</tr>
<tr>
<td>sssss</td>
<td>Starting track (CKD) or block (FBA) of disk extent</td>
</tr>
<tr>
<td>tttt</td>
<td>Number of tracks</td>
</tr>
<tr>
<td>vvvvvv</td>
<td>Volume serial number</td>
</tr>
</tbody>
</table>
The CA IDMS libraries, as established during installation

SYSIDMS  Filename of the SYSIDMS parameter file  
Note: For a complete listing of the SYSIDMS parameter file, see the CA IDMS Common Facilities Section.

userdb  Filename of the user database file

userdb  File-ID of the user database file

chain.file  File-ID of the chain file generated by IDMSDBN1

SYS004  Logical unit assignment of the sort work file

cuu  Physical device assignment of the sort work file

output. work  Output file-ID of the intermediate chain file used by IDMSDBN2

input.work  Input file-ID of the intermediate chain file used by IDMSDBN2

IDMSDIRL

Local mode IDMSDIRL (z/VSE)

// DLBL  idmslib,'idms.library'
// EXTENT SYSnnn,nnnnnn,,ssss,tttt
// ASSGN SYSnnn,DISK,VOL=vvvvv,SHR
// LIBDEF *,SEARCH=CA IDMS libraries
// EXEC PROC=IDMSLBSL
// TLBL SYSnnn,'idms.tapejrnl',,nnnnnn,,f
// ASSGN SYSnnn,x'cuu'
// DLBL  dictdb,'idms.dictdb',,DA
// EXTENT SYS005,nnnnnn
// ASSGN SYS005,DISK,VOL=nnnnnn,SHR
// TLBL SYS001,'idms.dirldata',,nnnnnn,,f
// ASSGN SYS001,x'cuu'
// EXEC  IDMSDIRL

Insert SYSIDMS parameters, as required
/*
IDMSDIRL input parameters
/*

IDMSLBSL  Name of the procedure provided at installation containing the file definitions for CA IDMS dictionaries, databases, and SYSIDMS parameter file.  
Note: For a complete listing of IDMSLBSL, see "IDMSLBSL Procedure" later in this section.

idmslib  Dtfname of the CA IDMS library

idms. library  Data set name of CA IDMS libraries, as established during installation

SYSnnn  Logical unit of the volume for which the extent is effective

nnnnnn  Volume serial identifier of appropriate disk volume
ssss  Starting track (CKD) or block (FBA) of disk extent

tttt  Number of tracks

vvvvv  Volume serial number

**CA IDMS libraries** The CA IDMS libraries, as established during installation

**SYSIDMS** Filename of the SYSIDMS parameter file

*Note:* For a complete listing of the SYSIDMS parameter file, see the **CA IDMS Common Facilities Section**.

**idms. tapejrnl** File-ID of the tape journal file

**f** File number of the tape file

**dictdb** Filename of the data dictionary file

**idms. dictdb** File-ID of the data dictionary file

**idms. dirldata** File-ID of the IDMSDIRL input file

**Central version**

To execute IDMSDIRL under the central version,

- Remove the TLBL and ASSGN statements for the tape journal file from the previous JCL
- Add a SYSCTL file

**IDMSLOOK**

**IDMSLOOK (z/VSE)**

```
// DLBL  idmslib,'idms.library'
// EXTENT  SYSnnn,nnn,ssss,tttt
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// TLBL  SYSnnn,'idms.tapejrnl',nnn,vvvvvv,f
// ASSGN  SYSnnn,'x'cuu'
// LIBDEF  *,SEARCH=CA IDMS libraries
// EXEC  PROC=IDMSBLLS
// EXEC  IDMSLOOK,SIZE=
```

Insert SYSIDMS parameters, as required

```
/*
Insert IDMSLOOK parameters
*/
```

**IDMSBLLS** Name of the procedure provided at installation containing the file definitions for CA IDMS dictionaries, databases, and SYSIDMS parameter file.

*Note:* For a complete listing of IDMSBLLS, see "IDMSBLLS Procedure" later in this section.

**idmslib** Dtfname of the CA IDMS library
**idms. library**
Data set name of CA IDMS libraries, as established during installation

**SYSnnn**
Logical unit of the volume for which the extent is effective

**nnnnnn**
Volume serial identifier of appropriate disk volume

**ssss**
Starting track (CKD) or block (FBA) of disk extent

**tttt**
Number of tracks

**vvvvv**
Volume serial number

**CA IDMS libraries**
The CA IDMS libraries, as established during installation

**SYSIDMS**
Filename of the SYSIDMS parameter file

*Note: For a complete listing of the SYSIDMS parameter file, see the CA IDMS Common Facilities Section.*

---

**IDMSRPTS**

**Local mode**

**IDMSRPTS (z/VSE)**

```
// DLBL   idmslib,'idms.library'
// EXTENT SYSnnn,nnnnnn,,,ssss,tttt
// ASSGN  SYSnnn,DISK,VOL=vvvvv,SHR
// LIBDEF *,SEARCH=CA IDMS libraries
// EXEC   PROC=IDMSLBSL
// TLBL   SYSnnn,'idms.tapejrnl',,nnnnnn,,f
// ASSGN  SYSnnn,x'cuu'
// DLBL   dictdb,'idms.dictdb',,DA
// EXTENT sys005,nnnnnn
// ASSGN  sys005,DISK,VOL=nnnnnn,SHR
// EXEC   IDMSRPTS
```

*Insert SYSIDMS parameters, as required*

/*
IDMSRPTS input parameters
/*

**IDMSLBSL**
Name of the procedure provided at installation containing the file definitions for CA IDMS dictionaries, databases, and SYSIDMS parameter file.

*Note: For a complete listing of IDMSLBSL, see "IDMSLBSL Procedure" later in this section.*

**idmslib**
Dtfname of the CA IDMS library

**idms. library**
Data set name of CA IDMS libraries, as established during installation

**SYSnnn**
Logical unit of the volume for which the extent is effective

**nnnnnn**
Volume serial identifier of appropriate disk volume

**ssss**
Starting track (CKD) or block (FBA) of disk extent
### CA IDMS Libraries

The CA IDMS libraries, as established during installation

### SYSIDMS

Filename of the SYSIDMS parameter file

**Note:** For a complete listing of the SYSIDMS parameter file, see the *CA IDMS Common Facilities Section*.

### SYS009

Logical unit assignment of the tape journal file

### idms. tapejrnl

File-id of the tape journal file

### nnnnnn

Volume serial number

### f

File number of the tape journal file

### dictdb

Filename of the data dictionary file

### idms. dictdb

File-id of the data dictionary file

### sys005

Logical unit assignment of the data dictionary file

---

### Central version

To execute IDMSRPTS under the central version,

- Remove the TLBL and ASSGN statements for the tape journal file from the previous JCL.

- Add a SYSCTL file

---

**IDMSRSTC**

### Local mode

**IDMSRSTC (z/VSE)**

```plaintext
// DLBL  idmslib,'idms.library'
// EXTENT SYSnnn,nnnnnn,,ssss,tttt
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// LIBDEF *,SEARCH=CA IDMS libraries
// EXEC PROC=IDMSLRLS
// TLBL  SYSnnn,'idms.tapejrnl',,nnnnnn,,f
// ASSGN  SYSnnn,'x'cuu'
// DLBL  dictdb,'idms.dictdb',A
// EXTENT  SYS005,nnnnnn
// ASSGN  SYS005,DISK,VOL=nnnnnn,SHR
// DLBL  IJSYSPH, 'rstc.macros',0
// EXTENT  SYSpch,nnnnnn,,ssss,,llll
// ASSGN  SYSpch,'x'cuu'
// EXEC  IDMSRSTC
```

Insert SYSIDMS parameters

/*
IDMSRSTC input parameters

/*

IDMSLBS  Name of the procedure provided at installation containing the file definitions for CA IDMS
dictionaries, databases, and SYSIDMS parameter file.

  Note: For a complete listing of IDMSLBS, see "IDMSLBS Procedure" later in this section.

idmslib  Dfname of the CA IDMS library

idms. library  Data set name of CA IDMS libraries, as established during installation

SYSnnn  Logical unit of the volume for which the extent is effective

nnnnnn  Volume serial identifier of appropriate disk volume

ssss  Starting track (CKD) or block (FBA) of disk extent

tttt  Number of tracks

vvvvv  Volume serial number

CA IDMS libraries  The CA IDMS libraries, as established during installation

SYSIDMS  Filename of the SYSIDMS parameter file

  Note: For a complete listing of the SYSIDMS parameter file, see the CA IDMS Common
  Facilities Section.

sys009  Logical unit assignment of the tape journal file

idms. tapejrnl  File-id of the tape journal file

nnnnnn  Volume serial number

f  File number of the tape journal file

dictdb  Filename of the data dictionary file

idms. dictdb  File-id of the data dictionary file

sys005  Logical unit assignment of the data dictionary file

rstc. macros  File-id of the card-image file containing the output IDMSRSTT macro statements

ssss  Starting track (CKD) or block (FBA) of the disk extent

llll  Number of tracks (CKD) or blocks (FBA) in the disk extent

cuu  Physical device assignment of the card-image file

Central version

To execute IDMSRSTC under the central version:

- Remove the TLBL and ASSGN statements for the tape journal from the previous JCL.

- Add a SYSCTL file
Assemble and link an IDMSRSTT module

// DLBL userlib
// EXTENT ,nnnnnn
// LIBDEF SOURCE,SEARCH=(userlib.idmslib)
// OPTION CATAL
// PHASE idmsrstt,*
// EXEC ASMA90

   Put IDMSRSTT macro statements here
/*
   Optionally add an INCLUDE statement for each procedure
   named in the NUPROCS= clause here. If not included, any
   procedures will be dynamically loaded.
*/

// DLBL userlib
// EXTENT ,nnnnnn
// LIBDEF PHASE,CATALOG=(userlib.idmslib)
// EXEC LNKEDT

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userlib</td>
<td>Filename of the user library</td>
</tr>
<tr>
<td>nnnnnn</td>
<td>Volume serial number of the library</td>
</tr>
<tr>
<td>userlib.idmslib</td>
<td>File identifier of the CA IDMS sublibrary</td>
</tr>
<tr>
<td>idmsrstt</td>
<td>Phase name of the IDMSRSTT table</td>
</tr>
</tbody>
</table>

**IDMSLBLS Procedure for z/VSE installation**

IDMSLBLS is a procedure provided during a z/VSE installation. It contains file definitions for the following CA IDMS components. These components are provided during installation:

- Dictionaries
- Sample databases
- Disk journal files
- SYSIDMS file

⚠️ **Note:** You can define a SYSCTL procedure that overrides IDMSOPTI specifications for central version operations. For more information, see the *CA IDMS Installation and Maintenance Section -- z/VSE*.

Tailor the IDMSLBLS procedure to reflect the filenames and definitions in use at your site and include this procedure in z/VSE JCL job streams.
The sample z/VSE JCL provided in this document includes the IDMSLBLS procedure. Therefore, individual file definitions for CA IDMS dictionaries, sample databases, disk journal files, and SYSIDMS file are not included in the sample JCL.

IDMSLBLS procedure listing

/* ------------------------ LABELS ------------------------*/
/* DLBL idmslib,'idms.library',1999/365 */
/* EXTENT ,nnnnnn,,ssss,1500 */
/* DLBL dccat,'idms.system.dccat',1999/365,DA */
/* EXTENT SYSnnn,nnnnnn,,ssss,31 */
/* ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR */
/* DLBL dccatl,'idms.system.dccatlod',1999/365,DA */
/* EXTENT SYSnnn,nnnnnn,,ssss,6 */
/* ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR */
/* DLBL dccatx,'idms.system.dccatx',1999/365,DA */
/* EXTENT SYSnnn,nnnnnn,,ssss,11 */
/* ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR */
/* DLBL dcdml,'idms.system.ddldml',1999/365,DA */
/* EXTENT SYSnnn,nnnnnn,,ssss,101 */
/* ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR */
/* DLBL dclod,'idms.system.ddldclod',1999/365,DA */
/* EXTENT SYSnnn,nnnnnn,,ssss,21 */
/* ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR */
/* DLBL dclog,'idms.system.ddldclog',1999/365,DA */
/* EXTENT SYSnnn,nnnnnn,,ssss,401 */
/* ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR */
/* DLBL dclScr,'idms.sysloc.ddlocscr',1999/365,DA */
/* EXTENT SYSnnn,nnnnnn,,ssss,401 */
/* ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR */
/* DLBL dirldb,'idms.sysdirl.ddldml',1999/365,DA */
/* EXTENT SYSnnn,nnnnnn,,ssss,281 */
/* ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR */
/* DLBL dirllod,'idms.sysdirl.ddldclog',1999/365,DA */
/* EXTENT SYSnnn,nnnnnn,,ssss,281 */
/* ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR */
/* DLBL dirldb,'idms.sysdirl.ddldml',1999/365,DA */
/* EXTENT SYSnnn,nnnnnn,,ssss,281 */
/* ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR */
/* DLBL dirllod,'idms.sysdirl.ddldclog',1999/365,DA */
/* EXTENT SYSnnn,nnnnnn,,ssss,281 */
/* ASSGN SYSnnn,DISK,VOL=vvvvvv,SHR */
```plaintext
// DLBL  secdd,'idms.sysuser.dlsec',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,sss,26
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL  dlddb,'idms.appdict.dlddml',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,sss,51
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL  dloddod,'idms.appdict.dloddclod',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,sss,51
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL  dlodd,'idms.appdict.dloddclod',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,sss,101
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL  dqld,'idms.syssql.dlqld',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,sss,201
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL  dqld,'idms.syssql.dlqld',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,sss,201
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL  ASFDEFN,'idms.asfdict.asfdefn',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,sss,101
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL  j1jrnl,'idms.j1jrnl',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,sss,54
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL  j2jrnl,'idms.j2jrnl',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,sss,54
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL  j3jrnl,'idms.j3jrnl',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,sss,54
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL  SYSIDMS,'#SYSIPT',0,SD
/+ */

<p>| Name of the sublibrary within the library containing CA IDMS modules | idmslib.sublib |
| Name of the sublibrary within the library containing user modules | user.sublib |
| Filename of the file containing CA IDMS modules | idmslib |
| File-ID associated with the file containing CA IDMS modules | idms.library |
| Logical unit of the volume for which the extent is effective | SYSnnn |
| Volume serial identifier of appropriate disk volume | nnnnnn |
| Starting track (CKD) or block (FBA) of disk extent | ssss |
| Filename of the system dictionary catalog (DDLCAT) area | dccat |
| File-ID of the system dictionary catalog (DDLCAT) area | idms.system.dccat |
| Filename of the system dictionary catalog load (DDLCATLOD) area | dccatl |
| File-ID of the system dictionary catalog load (DDLCATLOD) area | idms.system.dccatlod |
| Filename of the system dictionary catalog index (DDLCATX) area | dccax |
| File-ID of the system dictionary catalog index (DDLCATX) area | idms.system.dccatx |
| Filename of the system dictionary definition (DDLDMML) area | dcdml |
| File-ID of the system dictionary definition (DDLDMML) area | idms.system.dcdml |
| Filename of the system dictionary catalog load (DDLCATLOD) area | dcdml |
| File-ID of the system dictionary catalog load (DDLCATLOD) area | idms.system.dcdml |
| Filename of the system dictionary catalog index (DDLCATX) area | dcdx |
| File-ID of the system dictionary catalog index (DDLCATX) area | idms.system.dcdx |
| Filename of the system dictionary definition (DDLDMML) area | dcdy |
| File-ID of the system dictionary definition (DDLDMML) area | idms.system.dcdy |
| Filename of the system dictionary catalog load (DDLCATLOD) area | dcdzl |
| File-ID of the system dictionary catalog load (DDLCATLOD) area | idms.system.dcdzl |
| Filename of the system dictionary catalog index (DDLCATX) area | dcdx |
| File-ID of the system dictionary catalog index (DDLCATX) area | idms.system.dcdx |
| Filename of the system dictionary definition (DDLDMML) area | dcdy |
| File-ID of the system dictionary definition (DDLDMML) area | idms.system.dcdy |
| Filename of the system dictionary catalog load (DDLCATLOD) area | dcdzl |
| File-ID of the system dictionary catalog load (DDLCATLOD) area | idms.system.dcdzl |
| Filename of the system dictionary catalog index (DDLCATX) area | dcdx |
| File-ID of the system dictionary catalog index (DDLCATX) area | idms.system.dcdx |
| Filename of the system dictionary definition (DDLDMML) area | dcdy |
| File-ID of the system dictionary definition (DDLDMML) area | idms.system.dcdy |
| Filename of the system dictionary catalog load (DDLCATLOD) area | dcdzl |
| File-ID of the system dictionary catalog load (DDLCATLOD) area | idms.system.dcdzl |
| Filename of the system dictionary catalog index (DDLCATX) area | dcdx |
| File-ID of the system dictionary catalog index (DDLCATX) area | idms.system.dcdx |
| Filename of the system dictionary definition (DDLDMML) area | dcdy |
| File-ID of the system dictionary definition (DDLDMML) area | idms.system.dcdy |</p>
<table>
<thead>
<tr>
<th>Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>idms.system.ddldml</td>
<td>File-ID of the system dictionary definition (DDLDMML) area</td>
</tr>
<tr>
<td>dclod</td>
<td>Filename of the system dictionary definition load (DDLDCLOD) area</td>
</tr>
<tr>
<td>idms.system.ddldclog</td>
<td>File-ID of the system dictionary definition load (DDLDCLOD) area</td>
</tr>
<tr>
<td>dclod</td>
<td>Filename of the system log (DDLDCLOG) area</td>
</tr>
<tr>
<td>idms.system.ddldclog</td>
<td>File-ID of the system log (DDLDCLOG) area</td>
</tr>
<tr>
<td>dcrun</td>
<td>Filename of the system queue (DDLDCRUN) area</td>
</tr>
<tr>
<td>idms.system.ddldcun</td>
<td>File-ID of the system queue (DDLDCRUN) area</td>
</tr>
<tr>
<td>dscr</td>
<td>Filename of the system scratch (DDLDCSCR) area</td>
</tr>
<tr>
<td>idms.system.ddldcscr</td>
<td>File-ID of the system scratch (DDLDCSCR) area</td>
</tr>
<tr>
<td>dcmg</td>
<td>Filename of the system message (DDLDCMSG) area</td>
</tr>
<tr>
<td>idms.sysmsg.ddldcmsg</td>
<td>File-ID of the system message (DDLDCMSG) area</td>
</tr>
<tr>
<td>dclscr</td>
<td>Filename of the local mode system scratch (DDLOCSCR) area</td>
</tr>
<tr>
<td>idms.sysloc.ddlocscr</td>
<td>File-ID of the local mode system scratch (DDLOCSCR) area</td>
</tr>
<tr>
<td>dirldb</td>
<td>Filename of the IDMSDIRL definition (DDLDMML) area</td>
</tr>
<tr>
<td>idms.sysdirl.ddldml</td>
<td>File-ID of the IDMSDIRL definition (DDLDMML) area</td>
</tr>
<tr>
<td>dirllod</td>
<td>Filename of the IDMSDIRL definition load (DDLDCLOD) area</td>
</tr>
<tr>
<td>idms.sysdirl.dirllod</td>
<td>File-ID of the IDMSDIRL definition load (DDLDCLOD) area</td>
</tr>
<tr>
<td>empedemo</td>
<td>Filename of the EMPDEMO area</td>
</tr>
<tr>
<td>idms.empdemo1</td>
<td>File-ID of the EMPDEMO area</td>
</tr>
<tr>
<td>insdemo</td>
<td>Filename of the INSDEMO area</td>
</tr>
<tr>
<td>idms.insdemo1</td>
<td>File-ID of the INSDEMO area</td>
</tr>
<tr>
<td>orgdemo</td>
<td>Filename of the ORGDEMO area</td>
</tr>
<tr>
<td>idms.orgdemo1</td>
<td>File-ID of the ORGDEMO area</td>
</tr>
<tr>
<td>empldem</td>
<td>Filename of the EMPLDEMO area</td>
</tr>
<tr>
<td>idms.sqldemo.empldemo</td>
<td>File-ID of the EMPLDEMO area</td>
</tr>
<tr>
<td>infodem</td>
<td>Filename of the INFODEMO area</td>
</tr>
<tr>
<td>idms.sqldemo.infodemo</td>
<td>File-ID of the INFODEMO area</td>
</tr>
<tr>
<td>projdem</td>
<td>Filename of the PROJDEMO area</td>
</tr>
<tr>
<td>idms.projseg.projdemo</td>
<td>File-ID of the PROJDEMO area</td>
</tr>
<tr>
<td>indxedem</td>
<td>Filename of the INDXDEMO area</td>
</tr>
<tr>
<td>idms.sqldemo.indxedemo</td>
<td>File-ID of the INDXDEMO area</td>
</tr>
<tr>
<td>sysctl</td>
<td>Filename of the SYSCTL file</td>
</tr>
<tr>
<td>idms.sysctl</td>
<td>File-ID of the SYSCTL file</td>
</tr>
<tr>
<td>secdd</td>
<td>Filename of the system user catalog (DDLSEC) area</td>
</tr>
<tr>
<td>idms.sysuser.ddlsec</td>
<td>File-ID of the system user catalog (DDLSEC) area</td>
</tr>
<tr>
<td>dictdb</td>
<td>Filename of the application dictionary definition area</td>
</tr>
<tr>
<td>File Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>idms.appdict.ddldml</td>
<td>File-ID of the application dictionary definition (DDLDML) area</td>
</tr>
<tr>
<td>dloddb</td>
<td>Filename of the application dictionary definition load area</td>
</tr>
<tr>
<td>idms.appdict.ddldclod</td>
<td>File-ID of the application dictionary definition load (DDLDCLOD) area</td>
</tr>
<tr>
<td>sqldd</td>
<td>Filename of the SQL catalog (DDLCAT) area</td>
</tr>
<tr>
<td>idms.syssql.ddlcat</td>
<td>File-ID of the SQL catalog (DDLCAT) area</td>
</tr>
<tr>
<td>sqllod</td>
<td>Filename of the SQL catalog load (DDLCATL) area</td>
</tr>
<tr>
<td>idms.syssql.ddlcatl</td>
<td>File-ID of SQL catalog load (DDLCATL) area</td>
</tr>
<tr>
<td>sqlld</td>
<td>Filename of the SQL catalog index (DDLCATX) area</td>
</tr>
<tr>
<td>idms.syssql.ddlcatx</td>
<td>File-ID of the SQL catalog index (DDLCATX) area</td>
</tr>
<tr>
<td>asfdml</td>
<td>Filename of the asf dictionary definition (DDLDML) area</td>
</tr>
<tr>
<td>idms.asfdict.ddldml</td>
<td>File-ID of the asf dictionary definition (DDLDML) area</td>
</tr>
<tr>
<td>asflod</td>
<td>Filename of the asf dictionary definition load (ASFLOD) area</td>
</tr>
<tr>
<td>idms.asfdict.asflod</td>
<td>File-ID of the asf dictionary definition load (ASFLOD) area</td>
</tr>
<tr>
<td>asfdata</td>
<td>Filename of the asf data (ASFDATA) area</td>
</tr>
<tr>
<td>idms.asfdict.asfdata</td>
<td>File-ID of the asf data (ASFDATA) area</td>
</tr>
<tr>
<td>ASFDEFN</td>
<td>Filename of the asf data definition (ASFDEFN) area</td>
</tr>
<tr>
<td>idms.asfdict.asfdefn</td>
<td>File-ID of the asf data definition (ASFDEFN) area</td>
</tr>
<tr>
<td>j1jrnl</td>
<td>Filename of the first disk journal file</td>
</tr>
<tr>
<td>idms.j1jrnl</td>
<td>File-ID of the first disk journal file</td>
</tr>
<tr>
<td>j2jrnl</td>
<td>Filename of the second disk journal file</td>
</tr>
<tr>
<td>idms.j2jrnl</td>
<td>File-ID of the second disk journal file</td>
</tr>
<tr>
<td>j3jrnl</td>
<td>Filename of the third disk journal file</td>
</tr>
<tr>
<td>idms.j3jrnl</td>
<td>File-ID of the third disk journal file</td>
</tr>
<tr>
<td>SYSIDMS</td>
<td>Filename of the SYSIDMS parameter file</td>
</tr>
</tbody>
</table>
Sample CMS commands to run CA IDMS/DB utility statements and programs are presented in the following pages:

- **Batch Command Facility (CMS Commands) (see page 428)** - Presents common commands used to run the Batch Command Facility
- **Utility Statements (CMS Commands) (see page 429)** - Contains an alphabetical list of links to additional commands required to run each utility statement and program
- **Utility Programs (CMS Commands) (see page 452)** - Present utility programs for the Batch Command Facility

### Batch Command Facility (CMS Commands)

The following CMS commands are used to execute the Batch Command Facility.

When using the IDMSBCF program to execute a utility statement, code these commands along with the required statements for each of the utilities.

The file assignments for each utility statement and program are presented on subsequent pages in this section.

**Local mode IDMSBCF (CMS)**

FILEDEF SYSLST PRINTER
FI dcmeg DISK idms dsmsgdb fm (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn)

*Insert file assignments required by the utility statements*

FILEDEF sysjrnl TAP1 SL VOLID nnnnnn (RECFM VB LRECL lll BLKSIZE bbbb
FILEDEF SYSIDMS DISK sysidsms input a
FILEDEF SYSIPT DISK utility input a
GLOBAL LOADLIB idmslib dbalib
GLOBAL LOADLIB idmslib
OSRUN IDMSBCF

**Important!** Additional file assignments might be needed for the user catalog and the system dictionary.

**Note:** When executing local mode or batch-to-CV mode execs in CMS, filedefs for CDMSLIB loadlibs are optional. However, if omitted, and multiple EXECOS OSRUN statements are coded in the same EXEC, you must code the following CMS statement prior to the first EXECOS OSRUN statement in the EXEC:

SET STORECLR ENDCMD

You must also code the following CMS statement following the last EXECOS OSRUN statement:

SET STORECLR ENDSVC
<table>
<thead>
<tr>
<th><strong>idmslib</strong></th>
<th>Filename of the load library containing CA IDMS executable modules</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>dbalib</strong></td>
<td>Filename of the load library containing the DMCL and database name table load modules</td>
</tr>
<tr>
<td><strong>dcmsg</strong></td>
<td>DDname of the system message (DDLDCMSG) area</td>
</tr>
<tr>
<td><strong>idms</strong></td>
<td>File identifier of the system message (DDLDCMSG) area</td>
</tr>
<tr>
<td><strong>dmsgdb fm</strong></td>
<td></td>
</tr>
<tr>
<td><strong>ppp</strong></td>
<td>Page size of the database file</td>
</tr>
<tr>
<td><strong>nnn</strong></td>
<td>Number of pages in the database file</td>
</tr>
<tr>
<td><strong>sysjrnl</strong></td>
<td>DDname of the tape journal file</td>
</tr>
<tr>
<td><strong>nnnnn</strong></td>
<td>Volume serial number of the tape journal file</td>
</tr>
<tr>
<td><strong>lll</strong></td>
<td>Record length of the tape journal file</td>
</tr>
<tr>
<td><strong>rbbbb</strong></td>
<td>Block size of the tape journal file</td>
</tr>
</tbody>
</table>
| **SYSIDMS** | DDname of the parameter file provided by CA IDMS to specify runtime directives and operating system-dependent parameters.  
**Note:** For more information about the SYSIDMS parameter file, see the CA IDMS Common Facilities Section. |
| **sysidms** | File identifier of the file containing SYSIDMS parameters if applicable |
| **input a** | |
| **utility input** | File identifier of the file containing utility statements |
| **a** | |

**Central version IDMSBCF (CMS)**

FI dcmsg DISK idms dmsgdb fm (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn  
FILEDEF SYSCTL DISK sysctl file fm  
FILEDEF sysjrnl DUMMY  
FILEDEF SYSLST PRINTER  
GLOBAL LOADLIB idmslib dbalib  
GLOBAL LOADLIB idmslib  

*Insert file assignments required by the utility statements*  
FILEDEF SYSIPT DISK utility input a  
OSRUN IDMSBCF

<table>
<thead>
<tr>
<th><strong>idmslib</strong></th>
<th>Filename of the load library containing CA IDMS executable modules</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>utility input a</strong></td>
<td>File identifier of the file containing utility statements</td>
</tr>
</tbody>
</table>

**Utility Statements (CMS Commands)**

This page provides information about the various utility statements for Batch Command Facility.

The following topics are discussed on this page:

- **ARCHIVE JOURNAL** (see page 430)
- **ARCHIVE LOG** (see page 431)
ARCHIVE JOURNAL

FILEDEF commands for the batch command facility (CMS)

FILEDEF j1jrnl DISK j1jrnl jrnfile fm (RECFM F LRECL lll XTENT nnn
FILEDEF j2jrnl DISK j2jrnl jrnfile fm (RECFM F LRECL lll XTENT nnn

Additional journal file assignments, as required

FILEDEF archjrnl TAP1 SL VOLID nnnnnn (RECFM VB LRECL rrrr BLKSIZE bbbb
### j1jrnl
DDname of the first disk journal file, as defined in the DMCL

### j1jrnl jrnlfile fm
File identifier of the first disk journal file

### j2jrnl
DDname of the second disk journal file

### j2jrnl jrnlfile fm
File identifier of the second disk journal file

### lll
Page size of the journal file

### nnn
Number of pages in the journal file

### archjrnl
DDname of the tape archive file, as defined in the DMCL

### nnnnn
Volume serial number of the tape archive file

### rrrr
Length of the longest record in the tape archive file

### bbbb
Block size of the tape archive file, as defined in the DMCL

---

### ARCHIVE LOG

FILEDEF commands for the batch command facility (CMS)

```plaintext
FI dlogdb DISK idms dlogdb fm (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn
FI dmsg DISK idms dmsgdb fm (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn
FILEDEF sysjrnl DUMMY
FILEDEF SYS002 DISK archive dbfile fm (RECFM VB LRECL 280 BLKSIZE bbbb
```

<table>
<thead>
<tr>
<th>dlogdb</th>
<th>DDname of the system log area</th>
</tr>
</thead>
<tbody>
<tr>
<td>idms</td>
<td>File identifier of the system log area</td>
</tr>
<tr>
<td>dlogdb fm</td>
<td></td>
</tr>
<tr>
<td>ppp</td>
<td>Page size of the database file</td>
</tr>
<tr>
<td>nnn</td>
<td>Number of pages in the database file</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dmsg</th>
<th>DDname of the system message (DDLDCMSG) area</th>
</tr>
</thead>
<tbody>
<tr>
<td>idms</td>
<td>File identifier of the system message (DDLDCMSG) area</td>
</tr>
<tr>
<td>dmsgdb fm</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sysjrnl</th>
<th>DDname of the tape journal file defined in the DMCL module named in the IDMSNWKS subschema</th>
</tr>
</thead>
<tbody>
<tr>
<td>archive</td>
<td>File identifier of the archive log file</td>
</tr>
<tr>
<td>dbfile fm</td>
<td></td>
</tr>
</tbody>
</table>

| bbbb | Block size of the archive log file; must be greater than or equal to 284 (typically, equal to 4 plus a multiple of 280) |

---

### BACKUP

FILEDEF commands for the batch command facility (CMS)
FILEDEF `userdb` DISK `userdb dbfile` `fm`

Additional file assignments, as required

FILEDEF `SYS001` TAP1 SL VOLID `nnnnnn` (RECFM VB LRECL `llll` BLKSIZE `bbbb`

| `userdb` | DDname of the database file |
| `userdb dbfile` | File identifier of the database file |
| `nnnnnn` | Volume serial number of the tape backup file |
| `llll` | Record length of the tape backup file; must be the size of the largest page being backed up. |
| `bbbb` | Block size of the tape backup file; must be as large as the smaller of: The size of the largest page being backed up, plus 8 or 32,760 |

**BUILD**

FILEDEF commands for the batch command facility (CMS)

FILEDEF `userdb` DISK `userdb dbfile` `fm`
FILEDEF `SYS002` DISK `user load` `fm`
FILEDEF `SYS003` TAP1 SL VOLID `nnnnnn` (RECFM VB LRECL `llll` BLKSIZE `bbbb`
FILEDEF `SYS0SPCH` DISK `sort build` `fm`
FILEDEF `SORTMSG` PRINTER
FILEDEF `SORTWK01` DISK `sortwknn` file `fm`

| `userdb` | DDname of the database file |
| `userdb dbfile` | File identifier of the database file |
| `user load` | File identifier of the input (SYS002) file; for sizing information see discussion of SYS003 |
| `SYS003` | DDname of the output (SYS003) file |
| `nnnnnn` | Volume serial number of the SYS003 file |
| `llll` | Record size of the SYS003 file |
| `bbbb` | Block size of the SYS003 file |
| `sort build` | File identifier of the SYSPCH file |
| `sortwknn` | File identifier of the SORTWKnn file |

⚠️ **Note:** When running a complete BUILD, SYS002 and SYS003 must point to the *same* intermediate file. When running a stepped BUILD, SYS002 and SYS003 must point to a *different* intermediate file.
CLEANUP

FILEDEF commands for the batch command facility (CMS)

FILEDEF sysjrnl DISK tapejrnl jrnfile fm
FILEDEF userdb DISK userdb dbfile fm

Additional database file assignments, as required

<table>
<thead>
<tr>
<th>sysjrnl</th>
<th>DDname of the tape journal file</th>
</tr>
</thead>
<tbody>
<tr>
<td>tapejrnl jrnfile fm</td>
<td>File identifier of the tape journal file</td>
</tr>
<tr>
<td>userdb</td>
<td>DDname of the user database file</td>
</tr>
<tr>
<td>userdb dbfile fm</td>
<td>File identifier of the user database file</td>
</tr>
</tbody>
</table>

Central version

To execute CLEANUP SEGMENT under the central version, modify the CMS commands shown previously, as follows:

- Optionally remove any journal and database DD statements.
- Identify the DC/UCF system to be accessed by CLEANUP SEGMENT using the CVMACH and CVNUM SYSIDMS parameters.

⚠️ **Note:** For more information about the SYSIDMS parameter file, see the CA IDMS Common Facilities Section.

CONVERT CATALOG

FILEDEF commands for the batch command facility (CMS)

FILEDEF ddlcat DISK sydict ddlcat f(RECFM F LRECL ppp BLKSIZE ppp XTENT nnn)
FILEDEF ddlcatx DISK sydict ddlcatx f(RECFM F LRECL ppp BLKSIZE ppp XTENT nnn)
FILEDEF j1jrnl DISK j1jrnl jrnfile e(RECFM F LRECL ppp BLKSIZE ppp XTENT nnn)

| ddlcat        | DDname of the database file containing the DDLCAT area of the dictionary to be converted |
| ucat ddlcat f | File identifier of the database file containing the DDLCAT area of the dictionary to be converted |
| ddlcatx       | DDname of the database file containing the DDLCATX area of the dictionary to be converted |
| ucat ddlcatx f | File identifier of the database file containing the DDLCATX area of the dictionary to be converted |
| j1jrnl        | DDname of the first journal file, as defined in the DMCL |
| j1jrnl jrnfile e | File identifier of the first journal file |
Central version

To execute CONVERT CATALOG under the central version, modify the CMS commands shown previously, as follows:

- Optionally remove any journal and database DD statements
- Identify the DC/UCF system to be accessed using the CVMACH and CVNUM SYSDMS parameters

⚠️ Note: For more information about the SYSDMS parameter file, see the CA IDMS Common Facilities Section.

**CONVERT PAGE**

**FILEDEF commands for the batch command facility (CMS)**

FILEDEF `userdb` DISK `userdb dbfile fm`

*Additional existing database file assignments, as required*

FILEDEF `newdb` DISK `newdb dbfile fm`

*Additional converted database file assignments as required*

<table>
<thead>
<tr>
<th><code>userdb</code></th>
<th>DDname of the input database file</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>userdb dbfile fm</code></td>
<td>File identifier of the existing database file</td>
</tr>
<tr>
<td><code>newdb</code></td>
<td>DDname of the output converted database file</td>
</tr>
<tr>
<td><code>newdb dbfile fm</code></td>
<td>File identifier of the output converted database file</td>
</tr>
</tbody>
</table>

**EXPAND PAGE**

**FILEDEF commands for the batch command facility (CMS)**

FILEDEF `userdb` DISK `userdb dbfile fm`

*Additional existing database file assignments, as required*

FILEDEF `xfile` DISK `xbase dbfile fm` *(expanded-database-size)*

<table>
<thead>
<tr>
<th><code>userdb</code></th>
<th>DDname of the existing database file <em>(as specified by the FILE parameter)</em></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>userdb dbfile fm</code></td>
<td>File identifier of the existing database file</td>
</tr>
<tr>
<td><code>xfile</code></td>
<td>DDname of the expanded database file <em>(as specified by the FILEOUT parameter)</em></td>
</tr>
<tr>
<td><code>xfile dbfile fm</code></td>
<td>File identifier of the expanded database file</td>
</tr>
<tr>
<td></td>
<td>DCB information for the expanded database file</td>
</tr>
</tbody>
</table>
expanded-database-size

EXTRACT JOURNAL

FILEDEF commands for the batch command facility (CMS)

FILEDEF SYS001 DISK archive ft fmFILEDEF extract DISK extj (RECFM VB BLKSIZE bbbbb XTE NT nn
FILEDEF SORTMSG PRINTER
FILEDEF SORTWK01 DISK fn ft fm

Add additional SORTWKnn files as necessary

archive File identifier of the complete archive or journal file. It can be on tape or disk and can be
ft fm concatenated.
extract DDname of the extract journal file. If you do not specify one, it defaults to SYS002.
bbbbb Block size of the extract journal file. Specify a size at least as large as the largest block size
on the journal or archive files being processed.

FASTLOAD

FILEDEF commands for the batch command facility (CMS)

FILEDEF SYS001 DISK tblu sortparm fm
FILEDEF SYS002 DISK fmtpgm file fm
FILEDEF SYS004 TAP1 SL VOLID dbi004 (RECFM VB LRECL rr04 BLKSIZE bb04
FILEDEF SYS005 TAP2 SL VOLID dbi005 (RECFM VB LRECL rr05 BLK12E bb05
FILEDEF SYS009 TAP3 SL VOLID dbi009 (RECFM VB LRECL rr09 BLKSIZE bb09
FILEDEF SYS010 TAP4 SL VOLID dbi010 (RECFM VB LRECL rr10 BLKSIZE bb10
FILEDEF SYS011 TAP5 SL VOLID dbi011 (RECFM VB LRECL rr11 BLKSIZE bb11
FILEDEF SYSPCH DISK sortld file fm (RECFM F LRECL 80 BLKSIZE 80
FILEDEF RELDCTL DISK relctl ctl fm (RECFM FB LRECL 60 BLKSIZE bbbctl
FILEDEF SORTMSG PRINTER
FILEDEF SORTWK01 DISK sortwk file fm

Additional sort files, as required

tblu File identifier of the file containing sort parameters created by IDMSTBLU
sortparm fm
fmtpgm file File identifier of the file generated by the format program
fm
dbi004 Volume serial number of the intermediate work file containing the output from SORT1
rr04 Record size of the SYS004 file; should be the same as the record size for SYS001
bb04 Block size of the SYS004 intermediate work file; should be the same as the block size for SYS001
dbi005
<table>
<thead>
<tr>
<th>Description</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume serial number of the intermediate work file containing a control record and set membership information from IDMSDBL2</td>
<td>rr05</td>
</tr>
<tr>
<td>Record size of the SYS005 file</td>
<td>bb05</td>
</tr>
<tr>
<td>Block size of the SYS005 intermediate work file; must be at least rr05 plus four bytes</td>
<td>bb05</td>
</tr>
<tr>
<td>Volume serial number of the intermediate work file containing the sorted contents of SYS005</td>
<td>dbl009</td>
</tr>
<tr>
<td>Record size of the SYS009 file</td>
<td>rr09</td>
</tr>
<tr>
<td>Block size of the SYS009 intermediate work file; should be the same as rr05</td>
<td>bb09</td>
</tr>
<tr>
<td>Volume serial number of the intermediate work file containing pointer information from IDMSDBL3</td>
<td>dbl010</td>
</tr>
<tr>
<td>Record size of the SYS010 file</td>
<td>rr10</td>
</tr>
<tr>
<td>Block size of the SYS010 intermediate work file; must be at least 60 bytes (the size of the control record plus four bytes)</td>
<td>bb10</td>
</tr>
<tr>
<td>Volume serial number of the intermediate work file containing sorted pointer information from SORT4</td>
<td>dbl011</td>
</tr>
<tr>
<td>Record size of the SYS011 file</td>
<td>rr11</td>
</tr>
<tr>
<td>Block size of the SYS011 intermediate work file; should be the same as bb10</td>
<td>bb11</td>
</tr>
<tr>
<td>File identifier of the SYSPCH file</td>
<td>sortld file fm</td>
</tr>
<tr>
<td>File identifier of the RELDCTL file</td>
<td>reldctl ctl fm</td>
</tr>
<tr>
<td>Block size of the RELDCTL file. It should be a multiple of 60 with a maximum size of 32,760</td>
<td>bbbctl</td>
</tr>
<tr>
<td>File identifier of the SORTWK file</td>
<td>sortwk file fm</td>
</tr>
</tbody>
</table>

**FIX ARCHIVE**

FILEDEF commands for the batch command facility (CMS)

```plaintext
FILEDEF SYS001 TAP1 SL VOLID tjrnold (RECFM VB LRECL lll BLKSIZE bbbb
FILEDEF SYS002 TAP2 SL VOLID tjrnlfix (RECFM VB LRECL lll BLKSIZE bbbb
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume serial number of the input tape journal file</td>
<td>tjrnold</td>
</tr>
<tr>
<td>Volume serial number of the output tape journal file</td>
<td>tjrnlfix</td>
</tr>
<tr>
<td>Block size of the tape journal file as defined in the DMCL</td>
<td>bbbb</td>
</tr>
</tbody>
</table>

**FIX PAGE**

FILEDEF commands for the batch command facility (CMS)
**Central version**

To execute FIX PAGE under the central version, modify the CMS commands shown previously, as follows:

- Optionally remove any journal and database DD statements.
- Identify the DC/UCF system to be accessed by FIX PAGE using the CVMACH and CVNUM SYSIDMS parameters.

**Note:** For more information about the SYSIDMS parameter file, see the CA IDMS Common Facilities Section.

---

**FORMAT**

**FILEDEF commands for the batch command facility (CMS)**

To format a new or existing database file:

FILEDEF userdb DISK userdb dbfile fm (RECFM F LRECL lll XTENT nnn
Additional file assignments, as required

To format a new or existing disk journal file:

FILEDEF j1jrnl DISK j1jrnl jrnlfile fm (RECFM F LRECL lll XTENT nnn
Additional journal file assignments, as required

To format a new or existing SYSTRK file:

FILEDEF ffff DISK idms systrk1 fm (RECFM F LRECL lll XTENT nnn
Additional journal file assignments, as required

**userdb**  DDname of the user database file
**userdb dbfile fm**  File identifier of the user database file
**j1jrnl**  DDname of the first disk journal file, as defined in the DMCL
**j1jrnl jrnlfile fm**  DDname of the first disk journal file
**ffff**  DDname of the SYSTRK file
**idms systrk1 fm**  File identifier of the SYSTRK file
Central version

To execute FORMAT AREA or FORMAT SEGMENT under the central version, modify the CMS commands shown previously as follows:

- Optionally remove any journal and database DD statements.
- Identify the DC/UCF system to be accessed by FORMAT AREA or FORMAT SEGMENT using the CVMACH and CVNUM SYSIDMS parameters.

Note: For more information about the SYSIDMS parameter file, see the CA IDMS Common Facilities Section.

INSTALL STAMPS

Local Mode FILEDEF commands for the batch command facility (CMS)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILEDEF</td>
<td>DDname of the user database file</td>
<td>userdb DISK userdb dbfile fm</td>
</tr>
<tr>
<td>FILEDEF</td>
<td>File identifier of the user database file</td>
<td>userdb dbfile fm</td>
</tr>
<tr>
<td>FILEDEF</td>
<td>DDname of the database file containing the table</td>
<td>userdict DISK userdict dbfile fm</td>
</tr>
<tr>
<td></td>
<td>definitions</td>
<td>userdict dbfile fm</td>
</tr>
<tr>
<td>FILEDEF</td>
<td>DDname of the tape journal file</td>
<td>sysjrnl DISK sysjrnl jrnlfile fm</td>
</tr>
<tr>
<td></td>
<td>File identifier of the tape journal file</td>
<td>sysjrnl jrnlfile fm</td>
</tr>
</tbody>
</table>

Central Version

To execute INSTALL STAMPS under the central version, modify the JCL shown previously, as follows:

- Remove the USERDICT and SYSJRNL filedef statements
- Insert the following statement after the USERDB filedef statement:

  FILEDEF SYSCTL DISK sysctl file fm

<table>
<thead>
<tr>
<th>File identifier of the SYSCTL file</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysctl file fm</td>
</tr>
</tbody>
</table>
FILEDEF userdict DISK userdict dbfile fm
FILEDEF userdb DISK userdb dbfile fm
FILEDEF SYS001 DISK user input fm
FILEDEF SYS002 TAP1 SL VOLID loadin (RECFM VB LRECL rrin BLKSIZE bbin
FILEDEF SYS003 TAP2 SL VOLID lodout (RECFM VB LRECL rrout BLKSIZE bbout
FILEDEF SYSPCH DISK sortload file fm (RECFM F LRECL 80 BLKSIZE 80

The SORTMSG and SORTWKnn files are needed only when performing a complete LOAD.

FILEDEF SORTMSG PRINTER
FILEDEF SORTWK01 DISK sortwknn file fm

Additional sort files, as required

<table>
<thead>
<tr>
<th>userdict</th>
<th>DDname of the database file containing the dictionary with the table definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>userdict dbfile fm</td>
<td>File identifier of the database file containing the dictionary with the table definitions</td>
</tr>
<tr>
<td>userdb</td>
<td>DDname of the user database file</td>
</tr>
<tr>
<td>userdb dbfile fm</td>
<td>File identifier of the user database file</td>
</tr>
<tr>
<td>user input fm</td>
<td>File identifier of the input file</td>
</tr>
<tr>
<td>loadin*</td>
<td>Volume serial number of the input SYS002 file</td>
</tr>
<tr>
<td>rrin</td>
<td>Record size of the SYS002 file</td>
</tr>
<tr>
<td>bbin</td>
<td>Block size of the SYS002 file</td>
</tr>
<tr>
<td>lodout*</td>
<td>Volume serial number of the output SYS003 file</td>
</tr>
<tr>
<td>rrout</td>
<td>Record size of the SYS003 file</td>
</tr>
<tr>
<td>bbout</td>
<td>Block size of the SYS003 file</td>
</tr>
<tr>
<td>sortload file fm</td>
<td>File identifier of the SYSPCH file</td>
</tr>
<tr>
<td>sortwknn file fm</td>
<td>File identifier of the SORTWKnn file</td>
</tr>
</tbody>
</table>

Note: When running a complete LOAD, SYS002 and SYS003 must point to the same intermediate file. When running a stepped LOAD, SYS002 and SYS003 must point to a different intermediate file.

Note: When running a complete LOAD, you must preallocate the file referenced by SYS002 and SYS003 and do not use a temporary data set.

LOCK

FILEDEF commands for the batch command facility (CMS)

FILEDEF userdb DISK user userdb f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
Central version

To execute LOCK AREA under the central version, modify the CMS commands shown previously, as follows:

- Optionally remove any journal and database DD statements.
- Identify the DC/UCF system to be accessed by LOCK AREA using the CVMACH and CVNUM SYSIDMS parameters.

\[\text{Note: For more information about the SYSIDMS parameter file, see the CA IDMS Common Facilities Section.}\]

**MAINTAIN INDEX**

**FILEDEF commands for the batch command facility (CMS)**

```plaintext
FILEDEF userdb DISK usrrolddb dbfile fm
FILEDEF SYS003 TAP1 SL VOLID db1003 (RECFM VB LRECL rr03 BLKSIZE bb03
FILEDEF SYS004 TAP2 SL VOLID db1004 (RECFM VB LRECL rr04 BLKSIZE bb04
FILEDEF SYS005 TAP3 SL VOLID db1005 (RECFM VB LRECL rr05 BLKSIZE bb05
FILEDEF SYS006 TAP4 SL VOLID db1006 (RECFM VB LRECL rr06 BLKSIZE bb06
FILEDEF SYSPCH DISK sort file fm (RECFM F LRECL 80 BLKSIZE 80
FILEDEF RELDCTL DISK relcntl ctl fm (RECFM FB LRECL 60 BLKSIZE bbbctl
FILEDEF SORTMSG PRINTER
FILEDEF SORTWK01 DISK sortwk file fm
```

Additional sort files, as required

<table>
<thead>
<tr>
<th>userdb</th>
<th>DDname of the existing database file</th>
</tr>
</thead>
<tbody>
<tr>
<td>usrrolddb</td>
<td>File identifier of the existing database file</td>
</tr>
<tr>
<td>dbfile fm</td>
<td></td>
</tr>
<tr>
<td>dbl003</td>
<td>Volume serial number of the intermediate work file containing index descriptors from IDSMTABX</td>
</tr>
<tr>
<td>rr03</td>
<td>Record size of the SYS003 file; see MAINTAIN INDEX (see page 125) for sizing information</td>
</tr>
<tr>
<td>bb03</td>
<td>Block size of the SYS003 intermediate work file</td>
</tr>
<tr>
<td>dbl004</td>
<td>Volume serial number of the intermediate work file containing the output from SORT3</td>
</tr>
<tr>
<td>rr04</td>
<td>Record size of the SYS004 file; should be the same as rr03</td>
</tr>
<tr>
<td>bb04</td>
<td>Block size of the SYS004 intermediate work file; should be the same as bb03</td>
</tr>
<tr>
<td>dbl005</td>
<td>Volume serial number of the intermediate work file containing pointers for user-owned index sets from IDMSDBL3</td>
</tr>
</tbody>
</table>
**MERGE ARCHIVE**

**FILEDEF commands for the batch command facility (CMS)**

```plaintext
FILEDEF SYS001 TAP1 SL VOLID tjrnlold (RECFM F BLKSIZE bbbb)
FILEDEF SYS002 TAP2 SL VOLID tjrlfixi (RECFM F BLKSIZE bbbb)
FILEDEF JRNM01 TAP3 SL VOLID tmrgold (RECFM F BLKSIZE bbbb)
FILEDEF SORTMSG PRINTER
FILEDEF SORTWK01 DISK sortwk file fm

Additional sort files, as required
```

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tjrnlold</td>
<td>Volume serial number of the input tape journal file</td>
</tr>
<tr>
<td>tjrlfixi</td>
<td>Volume serial number of the output tape journal file</td>
</tr>
<tr>
<td>tmrgold</td>
<td>Volume serial number of the input tape merged journal file; if none, specify DUMMY</td>
</tr>
<tr>
<td>bbbb</td>
<td>Block size of the tape journal file as defined in the DMCL</td>
</tr>
</tbody>
</table>

**PRINT INDEX**

**FILEDEF commands for the batch command facility (CMS)**

```plaintext
FILEDEF sysjrnl DUMMY
FILEDEF userdb DISK userdb dbfile fm
```

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysjrnl</td>
<td>DDname of the dummy journal file</td>
</tr>
<tr>
<td>userdb</td>
<td>DDname of the database file</td>
</tr>
<tr>
<td>userdb dbfile fm</td>
<td>File identifier of the database file</td>
</tr>
</tbody>
</table>
PRINT JOURNAL

FILEDEF commands for the batch command facility (CMS)

FILEDEF SYS001 DISK archjrn1 jnlfile fm

<table>
<thead>
<tr>
<th>archjrn1</th>
<th>DDname of the archive journal file</th>
</tr>
</thead>
<tbody>
<tr>
<td>archjrn1 jnlfile.fm</td>
<td>File identifier of the archive journal file</td>
</tr>
</tbody>
</table>

PRINT LOG

FILEDEF commands for the batch command facility (CMS)

To print from the DDLDCLOG area:

FI dlogdb DISK idms dlogdb fm (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn

To print from the archive log file:

FI SYS001 DISK archive dbfile fm (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn
FI dcmsg DISK idms dmsgdb fm (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn
FILEDEF sysjrnl DUMMY

<table>
<thead>
<tr>
<th>dlogdb</th>
<th>DDname of the data dictionary log area</th>
</tr>
</thead>
<tbody>
<tr>
<td>idms dlogdb fm</td>
<td>File identifier of the data dictionary log area</td>
</tr>
<tr>
<td>archive dbfile fm</td>
<td>File identifier of the archive log file</td>
</tr>
<tr>
<td>ppp</td>
<td>Page size of the database file</td>
</tr>
<tr>
<td>nnn</td>
<td>Number of pages in the database file</td>
</tr>
<tr>
<td>dcmsg</td>
<td>DDname of the data dictionary message area</td>
</tr>
<tr>
<td>idms dmsgdb fm</td>
<td>File identifier of the data dictionary message area</td>
</tr>
<tr>
<td>sysjrnl</td>
<td>DDname of the tape journal file defined in the DMCL module named in the IDMSNWKS subschema</td>
</tr>
</tbody>
</table>

PRINT PAGE

FILEDEF commands for the batch command facility (CMS)

FILEDEF userdb1 DISK user userdb1 f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
FILEDEF userdb2 DISK user userdb2 f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
Additional database file assignments, as required

| userdb1 | DDname of the first database file |
Central version

To execute PRINT PAGE under the central version, modify the CMS commands shown previously, as follows:

- Optionally remove any journal and database DD statements.
- Identify the DC/UCF system to be accessed by PRINT PAGE using the CVMACH and CVNUM SYSIDMS parameters.

Note: For more information about the SYSIDMS parameter file, see the CA IDMS Common Facilities Section.

PRINT SPACE

FILEDEF commands for the batch command facility (CMS)

FILEDEF userdb1 DISK user userdb1 f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
FILEDEF userdb2 DISK user userdb2 f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn

Additional database file assignments, as required

Central version

To execute PRINT SPACE FOR AREA or PRINT SPACE FOR SEGMENT under the central version, modify the CMS commands shown previously as follows:

- Optionally remove any journal and database DD statements.
- Identify the DC/UCF system to be accessed by PRINT SPACE FOR AREA or PRINT SPACE FOR SEGMENT using the CVMACH and CVNUM SYSIDMS parameters.
Note: For more information about the SYSIDMS parameter file, see the CA IDMS Common Facilities Section.

PUNCH

Local Mode FILEDEF commands for the batch command facility (CMS)

FI usercat DISK user ddlcat fm (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
FI usercatl DISK user ddlcatl fm (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
FI usercatx DISK user ddlcatx fm (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
FILEDEF SYSPCH DISK punch fm
FILEDEF sysjrnl DUMMY

Additional dummied journals as required

<table>
<thead>
<tr>
<th>usercat</th>
<th>DDname of the DDLCAT area of the dictionary</th>
</tr>
</thead>
<tbody>
<tr>
<td>user ddlcat fm</td>
<td>File identifier of the DDLCAT area of the dictionary</td>
</tr>
<tr>
<td>usercatl</td>
<td>DDname of the DDLCATLOD area of the dictionary</td>
</tr>
<tr>
<td>user ddlcatl fm</td>
<td>File identifier of the DDLCATL area of the dictionary</td>
</tr>
<tr>
<td>usercatx</td>
<td>DDname of the DDLCATX area of the dictionary</td>
</tr>
<tr>
<td>user ddlcatx fm</td>
<td>File identifier of the DDLCATX area of the dictionary</td>
</tr>
<tr>
<td>sysjrnl</td>
<td>DDname of the tape journal file defined in the DMCL module named in the IDMSNWKS subschema</td>
</tr>
<tr>
<td>pppp</td>
<td>Page size of the database file</td>
</tr>
<tr>
<td>nnnn</td>
<td>Number of pages in the database file</td>
</tr>
<tr>
<td>punch fm</td>
<td>File identifier of the punch output</td>
</tr>
</tbody>
</table>

Central Version

To execute PUNCH under the central version, modify the JCL shown previously, as follows:

- Remove the USERCAT, USERCATL, USERCATX, and SYSJRNL statements
- Insert the following statement:
  FILEDEF SYSCTL DISK sysctl file fm

| sysctl file fm | File identifier of the SYSCTL file |
FILEDEF commands for the batch command facility (CMS)

**FILEDEF SYSO01 DISK** sortunld parms a
**FILEDEF SYSO02 DISK** user db1002 a
**FILEDEF SYSO03 DISK** user db1003 a
**FILEDEF SYSO04** tap4 SL VOLID nnnnnn (RECFM VB LRECL rrr004 BLKSIZE bbb004)
**FILEDEF SYSO05** tap5 SL VOLID nnnnnn (RECFM VB LRECL rrr005 BLKSIZE bbb005)
**FILEDEF SYSO06** tap6 SL VOLID nnnnnn (RECFM VB LRECL rrr006 BLKSIZE bbb006)
**FILEDEF SYSO07** tap7 SL VOLID nnnnnn (RECFM VB LRECL rrr007 BLKSIZE bbb007)
**FILEDEF SYSO08** tap8 SL VOLID nnnnnn (RECFM VB LRECL rrr008 BLKSIZE bbb008)
**FILEDEF SYSO09** tap9 SL VOLID nnnnnn (RECFM VB LRECL rrr009 BLKSIZE bbb009)
**FILEDEF SYSO10** tap10 SL VOLID nnnnnn (RECFM VB LRECL rrr010 BLKSIZE bbb010)
**FILEDEF SYSO11** tap11 SL VOLID nnnnnn (RECFM VB LRECL rrr011 BLKSIZE bbb011)
**FILEDEF SYS011 DISK** sortreld parms a
**FILEDEF relcntl DISK** user relcntl a (RECFM f LRECL 60 BLKSIZE bbbctl)
**FILEDEF usrdctl DISK** user usrdctl f (RECFM f LRECL pppp BLKSIZE pppp XTENT nnnn)

Additional database file assignments, as required

**FILEDEF SORTMSG PRINTER**
**FILEDEF SORTWK01 DISK** sortwk01 work a3

Additional sort files, as required

<table>
<thead>
<tr>
<th>sortunld parms a</th>
<th>File identifier of the sort parameters created by UNLOAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>user db1002 a</td>
<td>File identifier of the file generated by UNLOAD as SYSO01</td>
</tr>
<tr>
<td>user db1003 a</td>
<td>File identifier of the file generated by UNLOAD as SYSO03</td>
</tr>
<tr>
<td>tap4</td>
<td>Symbolic device name of the SYSO04 file containing the output from SORT1</td>
</tr>
<tr>
<td>rrr004</td>
<td>Record size of the SYSO04 file; should be the same as the record size for SYSO02 used by UNLOAD</td>
</tr>
<tr>
<td>bbb004</td>
<td>Block size of the SYSO04 intermediate work file; should be the same as the block size for SYSO02 used by UNLOAD</td>
</tr>
<tr>
<td>tap5</td>
<td>Symbolic device name of the SYSO05 file containing member descriptors for chained sets from IDMSDBL2</td>
</tr>
<tr>
<td>rrr005</td>
<td>Record size of the SYSO05 file</td>
</tr>
<tr>
<td>bbb005</td>
<td>Block size of the SYSO05 intermediate work file; must be at least 48 bytes (the size of a member descriptor plus four bytes)</td>
</tr>
<tr>
<td>tap6</td>
<td>Symbolic device name of the SYSO06 file containing member descriptors for user-owned index sets from IDMSDBL2</td>
</tr>
<tr>
<td>rrr006</td>
<td>Record size of the SYSO06 file</td>
</tr>
<tr>
<td>bbb006</td>
<td>Block size of the SYSO06 intermediate work file; must be at least 48 bytes (the size of a member descriptor plus four bytes)</td>
</tr>
<tr>
<td>tap7</td>
<td>Symbolic device name of the SYSO07 file containing the output of SORT2</td>
</tr>
<tr>
<td>rrr007</td>
<td></td>
</tr>
<tr>
<td><strong>Record size of the SYS007 file</strong>; should be the same as the larger of:</td>
<td></td>
</tr>
<tr>
<td>- the record size for SYS003 used by UNLOAD</td>
<td></td>
</tr>
<tr>
<td>- the record size for SYS006</td>
<td></td>
</tr>
<tr>
<td><strong>bbb007</strong></td>
<td>Block size of the SYS007 intermediate work file; must be at least as large as the larger of:</td>
</tr>
<tr>
<td>- the block size for SYS003 used by UNLOAD</td>
<td></td>
</tr>
<tr>
<td>- the block size for SYS006</td>
<td></td>
</tr>
<tr>
<td><strong>tap8</strong></td>
<td>Symbolic device name of the SYS008 file containing the reformatted index information from IDMSDBLX</td>
</tr>
<tr>
<td><strong>rrr008</strong></td>
<td>Record size of the SYS008 file; should be the same as rrr007</td>
</tr>
<tr>
<td><strong>bbb008</strong></td>
<td>Block size of the SYS008 intermediate work file; should be the same as bbb007</td>
</tr>
<tr>
<td><strong>tap9</strong></td>
<td>Symbolic device name of the SYS009 file containing the sorted index set descriptors from SORT3</td>
</tr>
<tr>
<td><strong>rrr009</strong></td>
<td>Record size of the SYS009 file; should be the same as the larger of:</td>
</tr>
<tr>
<td>- the record size for SYS005</td>
<td></td>
</tr>
<tr>
<td>- the record size for SYS008</td>
<td></td>
</tr>
<tr>
<td><strong>bbb009</strong></td>
<td>Block size of the SYS009 intermediate work file; must be at least as large as the larger of:</td>
</tr>
<tr>
<td>- the block size for SYS003 used by UNLOAD</td>
<td></td>
</tr>
<tr>
<td>- the block size for SYS006</td>
<td></td>
</tr>
<tr>
<td><strong>tap10</strong></td>
<td>Symbolic device name of the SYS010 file containing prefix pointer information from IDMSDBL3</td>
</tr>
<tr>
<td><strong>rrr010</strong></td>
<td>Record size of the SYS010 file</td>
</tr>
<tr>
<td><strong>bbb010</strong></td>
<td>Block size of the SYS010 intermediate work file; must be at least 44 bytes (the size of a pointer descriptor plus four bytes)</td>
</tr>
<tr>
<td><strong>tap11</strong></td>
<td>Symbolic device name of the SYS011 file containing the sorted prefix pointer information from SORT4</td>
</tr>
<tr>
<td><strong>rrr011</strong></td>
<td>Record size of the SYS011 file; should be the same as rrr010</td>
</tr>
<tr>
<td><strong>bbb011</strong></td>
<td>Block size of the SYS011 intermediate work file; should be the same as bbb010</td>
</tr>
<tr>
<td><strong>user relcntl a</strong></td>
<td>File identifier of the RELDCTL file containing control information created during the UNLOAD</td>
</tr>
<tr>
<td><strong>sortreld parms a</strong></td>
<td>File identifier of the SYSPCH file</td>
</tr>
<tr>
<td><strong>bbbbctl</strong></td>
<td>Blocksize of the RELDCTL file; it must be a multiple of 60 with a maximum size of 32,760</td>
</tr>
<tr>
<td><strong>sortwk01 work a3</strong></td>
<td>File identifier of temporary sort work file (if needed)</td>
</tr>
<tr>
<td><strong>userdb1</strong></td>
<td>DDname of the first database file</td>
</tr>
<tr>
<td><strong>user userdb1 f</strong></td>
<td>File identifier of the first database file</td>
</tr>
</tbody>
</table>

**RESTORE**

FILEDEF commands for the batch command facility (CMS)
FILEDEF userdb1 DISK user userdb1 f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn)
FILEDEF userdb2 DISK user userdb2 f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn)
FILEDEF SYS001 TAP1 SL VOLID vvvvvv (RECFM VB LRECL llll BLKSIZE bbbb)

Additional database file assignments, as required

<table>
<thead>
<tr>
<th>DDname of the first database file</th>
<th>File identifier of the first database file</th>
</tr>
</thead>
<tbody>
<tr>
<td>userdb1</td>
<td>user userdb1 f</td>
</tr>
<tr>
<td>user db2</td>
<td>DDname of the second database file</td>
</tr>
<tr>
<td>user userdb2 f</td>
<td>File identifier of the second database file</td>
</tr>
<tr>
<td>vvvvvv</td>
<td>Volume id of the tape file</td>
</tr>
<tr>
<td>llll</td>
<td>Record length of the tape backup file</td>
</tr>
<tr>
<td>bbbb</td>
<td>Block size of the tape backup file</td>
</tr>
</tbody>
</table>

**RESTRUCTURE**

**FILEDEF commands for the batch command facility (CMS)**

FILEDEF userdb1 DISK user userdb1 f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn)
FILEDEF userdb2 DISK user userdb2 f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn)
FILEDEF SYS001 DISK idms spill f (RECFM FB LRECL bbbb BLKSIZE bbbb XTENT nnnn)

<table>
<thead>
<tr>
<th>DDname of the first database file</th>
<th>File identifier of the first database file</th>
</tr>
</thead>
<tbody>
<tr>
<td>userdb1</td>
<td>user userdb1 f</td>
</tr>
<tr>
<td>user db2</td>
<td>DDname of the second database file</td>
</tr>
<tr>
<td>user userdb2 f</td>
<td>File identifier of the second database file</td>
</tr>
<tr>
<td>idms spill f</td>
<td>File identifier of the spill file</td>
</tr>
<tr>
<td>bbbb</td>
<td>Size of the spill file should be a multiple of 40 with a maximum size of 32,760</td>
</tr>
</tbody>
</table>

**RESTRUCTURE CONNECT**

**FILEDEF commands for the batch command facility (CMS)**

FILEDEF userdb DISK user userdb f
FILEDEF SYS001 DISK idms spill f
FILEDEF SORTMSG PRINTER
FILEDEF SORTWK01 DISK sortwk01 work a3

<table>
<thead>
<tr>
<th>DDname of the database file</th>
<th>File identifier of the database file</th>
</tr>
</thead>
<tbody>
<tr>
<td>userdb</td>
<td>user userdb f</td>
</tr>
<tr>
<td>user db</td>
<td>File identifier of the spill file</td>
</tr>
<tr>
<td>idms spill f</td>
<td></td>
</tr>
<tr>
<td>bbbb</td>
<td>Size of the spill file should be a multiple of 40 with a maximum size of 32,760</td>
</tr>
</tbody>
</table>
ROLLBACK

FILEDEF commands for the batch command facility (CMS)

FILEDEF SYS001 DISK tapn SL VOLID nnnnnn (RECFM VB LRECL 111 BLKSIZE bbbb
FILEDEF userdb DISK user userdb f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn

Additional database file assignments, as required

If using the SORT option add these statements:

FILEDEF SORTMSG PRINTER
FILEDEF SORTWK01 DISK sortwk01 disk a3

Additional sort files, as required

<table>
<thead>
<tr>
<th>tapn</th>
<th>Symbolic device name of the archive or tape journal file</th>
</tr>
</thead>
<tbody>
<tr>
<td>userdb</td>
<td>DDname of the user database file</td>
</tr>
<tr>
<td>user userdb f</td>
<td>File identifier of the user database file</td>
</tr>
</tbody>
</table>

ROLLFORWARD

FILEDEF commands for the batch command facility (CMS)

FILEDEF SYS001 DISK tapn SL VOLID nnnnnn (RECFM VB LRECL 111 BLKSIZE bbbb
FILEDEF userdb DISK user userdb f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn

Additional database file assignments, as required

If recovering from a journal extract file:

FILEDEF extract DISK jrnl.ext ft fm

If using the SORT option or processing a journal extract file, add these statements:

FILEDEF SORTMSG PRINTER
FILEDEF SORTWK01 DISK sortwk01 disk a3

Add additional sort files, as required

<table>
<thead>
<tr>
<th>tapn</th>
<th>Symbolic device name of the archive or tape journal file</th>
</tr>
</thead>
<tbody>
<tr>
<td>userdb</td>
<td>DDname of the user database file</td>
</tr>
<tr>
<td>user userdb f</td>
<td>File identifier of the user database file</td>
</tr>
<tr>
<td>extract</td>
<td>DDname of the extract journal file. If you don't specify, it defaults to SYS002.</td>
</tr>
<tr>
<td>jrnl.ext ft fm</td>
<td>File identifier of the journal extract file</td>
</tr>
</tbody>
</table>
SYNCHRONIZE STAMPS

Local Mode FILEDEF commands for the batch command facility (CMS)

FILEDEF userdb DISK userdb dbfile fm
FILEDEF userdict DISK userdict dbfile fm
FILEDEF sysjrnl DISK sysjrnl jrnlfie fm

<table>
<thead>
<tr>
<th>DDname</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userdb</td>
<td>DDname of the user database file</td>
</tr>
<tr>
<td>userdb dbfile fm</td>
<td>File identifier of the user database file</td>
</tr>
<tr>
<td>userdict</td>
<td>DDname of the database file containing the table definitions</td>
</tr>
<tr>
<td>userdict dbfile fm</td>
<td>File identifier of the database file containing the table definitions</td>
</tr>
<tr>
<td>sysjrnl</td>
<td>DDname of the tape journal file</td>
</tr>
<tr>
<td>sysjrnl jrnlfie fm</td>
<td>File identifier of the tape journal file</td>
</tr>
</tbody>
</table>

Central Version

To execute SYNCHRONIZE STAMPS under the central version, modify the JCL shown previously, as follows:

- Remove the USERDICT and SYSJRNL filedef statements
- Insert the following statement after the USERDB filedef statement:
  
  FILEDEF SYSCTL DISK sysctl file fm

<table>
<thead>
<tr>
<th>DDname</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysctl file fm</td>
<td>File identifier of the SYSCTL file</td>
</tr>
</tbody>
</table>

TUNE INDEX

FILEDEF commands for the batch command facility (CMS)

FILEDEF userdb DISK userdb dbfile fm

<table>
<thead>
<tr>
<th>DDname</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userdb</td>
<td>DDname of the database file</td>
</tr>
<tr>
<td>userdb dbfile fm</td>
<td>File identifier of the database file</td>
</tr>
</tbody>
</table>

Central version

To execute TUNE INDEX under the central version, modify the CMS commands shown previously, as follows:

- Optionally remove any journal and database DD statements.
- Identify the DC/UCF system to be accessed by TUNE INDEX using the CVMACH and CVNUM SYSIDMS parameters.
Note: For more information about the SYSIDMS parameter file, see the CA IDMS Common Facilities Section.

**UNLOCK**

FILEDEF commands for the batch command facility (CMS)

FILEDEF userdb DISK user userdb f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn)

<table>
<thead>
<tr>
<th>userdb</th>
<th>DDname of the database file</th>
</tr>
</thead>
<tbody>
<tr>
<td>user userdf</td>
<td>File identifier of the database file</td>
</tr>
</tbody>
</table>

**UNLOAD**

FILEDEF commands for the batch command facility (CMS)

FILEDEF userdb DISK user olddb f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn)

Additional database file assignments, as required

FILEDEF sysjrn1 DUMMY
FILEDEF SYS002 tap2 SL VOLID nnnnnn (RECFM VB LRECL rrr002 BLKSIZE bbb002)
FILEDEF SYS003 tap2 SL VOLID nnnnnn (RECFM VB LRECL rrr003 BLKSIZE bbb003)
FILEDEF SYSPCH DISK sort unld a (RECFM F BLKSIZE 80)
FILEDEF RELDCTL DISK user reldctl a (RECFM FB LRECL 60 BLKSIZE bbbctl)

<table>
<thead>
<tr>
<th>userdb</th>
<th>DDname of the existing database file</th>
</tr>
</thead>
<tbody>
<tr>
<td>user olddb</td>
<td>File identifier of the existing database file</td>
</tr>
<tr>
<td>sysjrn1</td>
<td>DDname of the dummy journal file</td>
</tr>
<tr>
<td>tap2</td>
<td>Symbolic device name of the SYS002 output file</td>
</tr>
<tr>
<td>rrr002</td>
<td>Record size of the SYS002 output file</td>
</tr>
<tr>
<td>bbb002</td>
<td>Block size of the SYS002 output file; must be at least the size of the largest database record plus 4 bytes</td>
</tr>
<tr>
<td>tap3</td>
<td>Symbolic device name of the SYS003 output file</td>
</tr>
<tr>
<td>rrr003</td>
<td>Record size of the SYS003 output file</td>
</tr>
<tr>
<td>bbb003</td>
<td>Block size of the SYS003 output file</td>
</tr>
<tr>
<td>sort unload</td>
<td>Data set name of the sort parameters created by UNLOAD</td>
</tr>
<tr>
<td>disk</td>
<td>Symbolic device name of the sort parameter file</td>
</tr>
<tr>
<td>bbbctl</td>
<td>Size of the RELDCTL file; it should be a multiple of 60 with a maximum size of 32,760</td>
</tr>
</tbody>
</table>
UPDATE STATISTICS

FILEDEF commands for the batch command facility (CMS)

FILEDEF userdb DISK user userdb f
Additional database file assignments, as required
FILEDEF ddlcat DISK sysdict ddlcat f (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn)
FILEDEF j1jrnl DISK j1jrnl jrnlfile e (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn)

Additional journal file assignments, as required

<table>
<thead>
<tr>
<th>userdb</th>
<th>DDname of the database file</th>
</tr>
</thead>
<tbody>
<tr>
<td>user userdb f</td>
<td>File identifier of the database file</td>
</tr>
<tr>
<td>ddlcat</td>
<td>DDname of the database file containing the area of the system dictionary with the table definitions</td>
</tr>
<tr>
<td>sysdict</td>
<td>File identifier of the database file containing the area of the system dictionary with the table definitions</td>
</tr>
<tr>
<td>ddlcat f</td>
<td>DDname of the database file containing the area of the system dictionary with indexes</td>
</tr>
<tr>
<td>sysdict</td>
<td>File identifier of the database file containing the area of the system dictionary with indexes</td>
</tr>
<tr>
<td>j1jrnl</td>
<td>DDname of the first journal file, as defined in the DMCL</td>
</tr>
<tr>
<td>j1jrnl jrnlfile e</td>
<td>File identifier of the first journal file</td>
</tr>
</tbody>
</table>

Central version

To execute UPDATE STATISTICS under the central version, modify the CMS commands shown previously as follows:

- Optionally remove any journal and database DD statements.
- Identify the DC/UCF system to be accessed by UPDATE STATISTICS using the CVMACH and CVNUM SYSIDMS parameters.

Note: For more information about the SYSIDMS parameter file, see the CA IDMS Common Facilities Section.

VALIDATE

FILEDEF commands for the batch command facility (CMS)

FILEDEF userdb DISK user userdb f (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn)
FILEDEF SYSP002 tap2 SL VOLID nnnnnn (RECFM VB LRECL rrrin BLKSIZE bbin)
FILEDEF SYSP003 tap3 SL VOLID nnnnnn (RECFM VB LRECL rrrout BLKSIZE bbbout)
FILEDEF SYSPCH DISK sort build fm
The SORTMSG and SORTWKnn files are needed only when performing a complete VALIDATE.

FILEDEF SORTMSG PRINTER
FILEDEF sortwknn DISK sort worknn a3

Additional sort files, as required

<table>
<thead>
<tr>
<th>DDname</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userdb</td>
<td>DDname of the database file</td>
</tr>
<tr>
<td>user userdb b</td>
<td>File identifier of the database file</td>
</tr>
<tr>
<td>tap2*</td>
<td>Symbolic device name of the input SYS002 file</td>
</tr>
<tr>
<td>rrin</td>
<td>Record size of the SYS002 file</td>
</tr>
<tr>
<td>bbbin</td>
<td>Block size of the SYS002 file</td>
</tr>
<tr>
<td>tap3*</td>
<td>Symbolic device name of the output SYS003 file</td>
</tr>
<tr>
<td>rrrout</td>
<td>Record size of the SYS003 file</td>
</tr>
<tr>
<td>bbbout</td>
<td>Block size of the SYS003 file</td>
</tr>
<tr>
<td>sort build fm</td>
<td>File identifier of the SYSPCH file</td>
</tr>
<tr>
<td>sortwknn file fm</td>
<td>File identifier of the SORTWKnn file</td>
</tr>
</tbody>
</table>

⚠️ **Note:** When running a complete VALIDATE, SYS002 and SYS003 must point to the same intermediate file. When running a stepped VALIDATE, SYS002 and SYS003 must point to a different intermediate file.

Utility Programs (CMS Commands)

The following utility programs for the Batch Command Facility are discussed on this page:

- IDMSDBAN (see page 452)
- IDMSDIRL (see page 454)
- IDMSLOOK (see page 455)
- IDMSRPTS (see page 455)
- IDMSRSTC (see page 456)
- IDMSRSTT (see page 457)

**IDMSDBAN**

FILEDEF userdb DISK user userdb b (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn)

Additional database file assignments, as required

FILEDEF SYS002 TAP1 SL VOLID nnnn (RECFM VB LRECL 512 BLKSIZE 9000)
FILEDEF SYSOUT PRINTER
FILEDEF SYSLST PRINTER
FILEDEF SYSIDMS DISK sysidms input a
FILEDEF SYSIPT DISK dban input a
GLOBAL LOADLIB idmslib dbalib
GLOBAL LOADLIB idmslib
OSRUN IDMSDBN1

FILEDEF SYS001 TAP1 SL VOLID nnnnn (RECFM VB LRECL 512 BLKSIZE 9000
FILEDEF SYS002 DISK chain work a3, (RECFM VB LRECL 512 BLKSIZE 9000
FILEDEF SYSOUT PRINTER
FILEDEF SYSLST PRINTER
FILEDEF SYSIDMS DISK sysidms input a
GLOBAL LOADLIB idmslib dbalib
GLOBAL LOADLIB idmslib
OSRUN IDMSDBN2

Important! Additional file assignments might be needed for the user catalog and the system dictionary.

<table>
<thead>
<tr>
<th>userdb</th>
<th>DDname of the user database file</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>File identifier of the user database file</td>
</tr>
<tr>
<td>userdb b</td>
<td></td>
</tr>
<tr>
<td>pppp</td>
<td>Page size of the user database file</td>
</tr>
<tr>
<td>nnnn</td>
<td>Number of pages in the user database file</td>
</tr>
<tr>
<td>nnnnnn</td>
<td>Volume serial number of the chain file generated by IDMSDBN1</td>
</tr>
<tr>
<td>SYSIDMS</td>
<td>DDname of the parameter file provided by CA IDMS to specify runtime directives and operating system-dependent parameters.</td>
</tr>
<tr>
<td>Note:</td>
<td>For more information about the SYSIDMS parameter file, see the CA IDMS Common Facilities Section.</td>
</tr>
<tr>
<td>sysidms</td>
<td>File identifier of the file containing SYSIDMS parameters if applicable</td>
</tr>
<tr>
<td>input a</td>
<td></td>
</tr>
<tr>
<td>dban input</td>
<td>File identifier of the file containing IDMSDBAN input parameters</td>
</tr>
<tr>
<td>a</td>
<td></td>
</tr>
<tr>
<td>dbalib</td>
<td>Filename of the load library containing the DMCL and database name table load modules</td>
</tr>
<tr>
<td>idmslib</td>
<td>Filename of the CA IDMS/DB load library containing the subschema and DMCL load modules</td>
</tr>
<tr>
<td>chain work a3</td>
<td>File identifier of the intermediate chain file used by IDMSDBN2</td>
</tr>
</tbody>
</table>

Note: IDMSDBAN requires the presence of an external sort package (other than the CMS SORT command) that can be loaded dynamically.
IDMSDIRL

Local mode IDMSDIRL (CMS)

FILEDEF sysjrnl TAP1 SL VOLID nnnnnn (RECFM VB LRECL lll BLKSIZE bbbb
FILEDEF dictdb DISK idsms dictdb f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
FILEDEF SYSLST PRINTER
FILEDEF SYS001 TAP2 SL VOLID nnndrl (RECFM VB LRECL 600 BLKSIZE 5992
FILEDEF SYSIDMS DISK sysidms input a
FILEDEF SYSIPT DISK dirl input a
GLOBAL LOADLIB idmslib dbalib
GLOBAL LOADLIB idmslib
OSRUN IDMSDIRL

Important! User catalog and the system dictionary depending on your security implementation.

<table>
<thead>
<tr>
<th>sysjrnl</th>
<th>DDname of the tape journal file, as defined in the DMCL module</th>
</tr>
</thead>
<tbody>
<tr>
<td>nnnnnn</td>
<td>Volume serial number of the tape journal file</td>
</tr>
<tr>
<td>lll</td>
<td>Record length of the tape journal file</td>
</tr>
<tr>
<td>bbbb</td>
<td>Block size of the tape journal file</td>
</tr>
<tr>
<td>dictdb</td>
<td>DDname of the data dictionary file</td>
</tr>
<tr>
<td>idsms</td>
<td>File identifier of the data dictionary file</td>
</tr>
<tr>
<td>dictdb f</td>
<td></td>
</tr>
<tr>
<td>pppp</td>
<td>Page size of the data dictionary file</td>
</tr>
<tr>
<td>nnnn</td>
<td>Number of pages in the data dictionary file</td>
</tr>
<tr>
<td>nnndrl</td>
<td>Volume serial number of the IDMSDIRL input file (on the installation media)</td>
</tr>
</tbody>
</table>

SYSIDMS DDname of the parameter file provided by CA IDMS to specify runtime directives and operating system-dependent parameters.  
Note: For more information about the SYSIDMS parameter file, see the CA IDMS Common Facilities Section.

<table>
<thead>
<tr>
<th>sysidms</th>
<th>File identifier of the file containing SYSIDMS parameters if applicable</th>
</tr>
</thead>
<tbody>
<tr>
<td>input a</td>
<td></td>
</tr>
<tr>
<td>dirl input a</td>
<td>File identifier of the file containing the IDMSDIRL input parameters</td>
</tr>
<tr>
<td>dbalib</td>
<td>Filename of the load library containing the DMCL and database name table load modules</td>
</tr>
<tr>
<td>idmslib</td>
<td>Filename of the CA IDMS/DB load library</td>
</tr>
</tbody>
</table>

Central version

To execute IDMSDIRL under the central version, modify the CMS commands shown previously, as follows:

- Remove the SYSJRNL and DICTDB FILEDEF commands.
Identify the DC/UCF system to be accessed by IDMSDIRL using the CVMACH and CVNUM SYSIDMS parameters.

IDMSLOOK

IDMSLOOK (CMS)

FILEDEF idmslib DISK idmslib loadlib a
FILEDEF SYSLST PRINTER
FILEDEF SYSIDMS DISK sysidms input a
FILEDEF SYSIPT look input a
GLOBAL LOADLIB idmslib dbalib
GLOBAL LOADLIB idmslib
OSRUN IDMSLOOK

Important! Additional file assignments might be needed for the user catalog and the system dictionary.

dbalib Filename of the load library containing the DMCL and database name table load modules
idmslib DDname for statement containing the CA IDMS load library containing CA IDMS modules
idmslib loadlib a File identifier of the CA IDMS load library containing CA IDMS modules
SYSIDMS DDname of the parameter file provided by CA IDMS to specify runtime directives and operating system-dependent parameters.
Note: For more information about the SYSIDMS parameter file, see the CA IDMS Common Facilities Section.
sysidms input a File identifier of the file containing SYSIDMS parameters, if applicable
look input a File identifier for the file containing the IDMSLOOK input parameters

IDMSRPTS

Local mode IDMSRPTS (CMS)

FILEDEF sysjrnl TAPI1 SL VOLID nnnnnn (RECFM VB LRECL lll BLKSIZE bbbb
FILEDEF dictdb DISK idms dictdb f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
FILEDEF SYSLST PRINTER
FILEDEF SYSIPT DISK sysidms input a
FILEDEF SYSIPT DISK rpts input a
GLOBAL LOADLIB idmslib dbalib
GLOBAL LOADLIB idmslib
OSRUN IDMSRPTS
### Important!
Additional file assignments might be needed for the user catalog and the system dictionary.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sysjrnl</code></td>
<td>DDname of the tape journal file, as defined in the DMCL module</td>
</tr>
<tr>
<td><code>nnnnnn</code></td>
<td>Volume serial number of the tape journal file</td>
</tr>
<tr>
<td><code>lll</code></td>
<td>Record length of the tape journal file</td>
</tr>
<tr>
<td><code>bbbb</code></td>
<td>Block size of the tape journal file, as defined in the DMCL module</td>
</tr>
<tr>
<td><code>dictdb</code></td>
<td>DDname of the data dictionary file</td>
</tr>
<tr>
<td><code>idms dictdb f</code></td>
<td>File identifier of the data dictionary file</td>
</tr>
<tr>
<td><code>pppp</code></td>
<td>Page size of the data dictionary file</td>
</tr>
<tr>
<td><code>nnnn</code></td>
<td>Number of pages in the data dictionary file</td>
</tr>
<tr>
<td><code>SYSIDMS</code></td>
<td>DDname of the parameter file provided by CA IDMS to specify runtime directives and operating system-dependent parameters.</td>
</tr>
</tbody>
</table>

**Note:** For more information about the SYSIDMS parameter file, see the CA IDMS Common Facilities Section.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sysidms input a</code></td>
<td>File identifier of the file containing SYSIDMS parameters, if applicable</td>
</tr>
<tr>
<td><code>rpts input a</code></td>
<td>File identifier of the file containing the IDMSRPTS input parameters</td>
</tr>
<tr>
<td><code>dbalib</code></td>
<td>Filename of the load library containing the DMCL and database name table load modules</td>
</tr>
<tr>
<td><code>idmslib</code></td>
<td>Filename of the CA IDMS/DB load library</td>
</tr>
</tbody>
</table>

### Central version

To execute IDMSRPTS under the central version, modify the CMS commands shown previously, as follows:

- Remove the SYSJRNL and DICTDB FILEDEF commands.
- Identify the DC/UCF system to be accessed by IDMSRPTS using the CVMACH and CVNUM SYSIDMS parameters.

---

**IDMSRSTC**

### Local mode IDMSRSTC (CMS)

```
FILEDEF sysjrnl TAP1 SL VOLID nnnnnn (RECFM VB LRECL lll BLKSIZE bbbb
FILEDEF dictdb DISK idms dictdb f (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
FILEDEF SYSLST PRINTER
FILEDEF SYSPCH DISK rsttmac ASSEMBLE A
FILEDEF SYSIDMS DISK sysidms input a
FILEDEF SYSIPT DISK sttc input a
GLOBAL LOADLIB idmslib dbalib
GLOBAL LOADLIB idmslib
OSRUN IDMSRSTC
```
Important! Additional file assignments might be needed for the user catalog and the system dictionary.

<table>
<thead>
<tr>
<th>Important! Additional file assignments might be needed for the user catalog and the system dictionary.</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysjrnl</td>
</tr>
<tr>
<td>nnnnnn</td>
</tr>
<tr>
<td>lll</td>
</tr>
<tr>
<td>bbbb</td>
</tr>
<tr>
<td>dictdb</td>
</tr>
<tr>
<td>idms dictdb</td>
</tr>
<tr>
<td>pppp</td>
</tr>
<tr>
<td>nnnn</td>
</tr>
<tr>
<td>rsttmac</td>
</tr>
<tr>
<td>ASSEMBLE A</td>
</tr>
<tr>
<td>SYSIDMS</td>
</tr>
<tr>
<td>Note: For more information about the SYSIDMS parameter file, see the CA IDMS Common Facilities Section.</td>
</tr>
<tr>
<td>sysidms input</td>
</tr>
<tr>
<td>rstc input</td>
</tr>
<tr>
<td>dbalib</td>
</tr>
<tr>
<td>idmslib</td>
</tr>
</tbody>
</table>

Central version

To execute IDMSRSTC under the central version, modify the CMS commands shown previously, as follows:

- Remove the SYSJRNL and DICTDB FILEDEF commands.
- Identify the DC/UCF system to be accessed by IDMSRTC using the CVMACH and CVNUM SYSIDMS parameters.

IDMSRSTT

Assemble and link an IDMSRSTT module

GLOBAL MACLIB idmslib
ASMAHL rsttmac (RENT

Optionally add the rstt module into a txtlib.
TX TLIB ADD userlib idmsrstt

Optionally link the rstt module into a loadlib.

FILEDEF idmsrsttt DISK idmsrsttt TEXT A

If linking NUPROCS= procedures from TEXT files
add a FILEDEF for each procedure. If linking from
a TX TLIB add a filedef for the txtlib.

Note: linking procedures is not required.
They will be dynamically loaded if not linked.

FILEDEF SYSLMOD DISK idmsload LOADLIB A (RECFM V LRECL 1024 BLKSIZE 1024
LKED linkctl (RMODE ANY LIST MAP XREF PRINT

Linkage editor control statements in linkctl file:

INCLUDE idmsrsttt

Optionally add an INCLUDE statement for each procedure
named in the NUPROCS= clause here.
If not included any procedures will be dynamically loaded.

NAME idmsrsttt(R)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>idmsrsttt</td>
<td>Module name of the IDMSRSTT table</td>
</tr>
<tr>
<td>rstttmac</td>
<td>Filename of the card-image file containing the IDMSRSTT macro statement</td>
</tr>
<tr>
<td>userlib</td>
<td>Filename of the user text library</td>
</tr>
<tr>
<td>idmslib</td>
<td>Filename of the library containing the IDMSRSTT macro</td>
</tr>
<tr>
<td>idmsload</td>
<td>Filename of the loadlib to store the rstt module</td>
</tr>
<tr>
<td>linkctl</td>
<td>Filename of TEXT file used as input to LKED</td>
</tr>
</tbody>
</table>
This section shows a listing of a format program that can be used with IDMSDBLU to load the sample Commonweather database provided during the CA IDMS/DB installation. The program is written in COBOL and has been run through the CA IDMS DML COBOL precompiler.

Format program for FASTLOAD

```
*RETRIEVAL
*DMLIST
IDENTIFICATION DIVISION.
PROGRAM-ID. EMPFLOAD.
*AUTHOR. KGV.
*
*INSTALLATION. CA
* 8600 BRYN MAWR AVENUE
* CHICAGO, IL 60131.
*
*DATE-WRITTEN. 08/20/90.
*UPDATED FOR 15.0. 11/16/00. LRD.
*
*REMARKS. THIS PROGRAM CREATES DATA TO
* BE USED AS INPUT TO THE FASTLOAD
* UTILITY, TO LOAD THE EMPLOYEE
* DEMO DATABASE. IT USES THE SAME
* INPUT AS EMPLOAD.

ENVIRONMENT DIVISION.
IDMS-CONTROL SECTION.
PROTOCOL. MODE IS BATCH DEBUG
IDMS-RECORDS WITHIN WORKING-STORAGE SECTION.

DATA DIVISION.
SCHEMA SECTION.
DB EMPSS01 WITHIN EMPSCHM VERSION 100.
SKIP2

WORKING-STORAGE SECTION.

01 OWNER-DESCRIPTOR-ONE.
 03 OWNER-ONE-SET PIC X(16).
 03 OWNER-ONE-serial PIC S9(8) COMP.
 03 OWNER-ONE-KEY PIC X(40).
 03 OWNER-ONE-KEY-RDEF REDEFINES OWNER-ONE-KEY.
      05 OWNER-ONE-KEY-SERIAL PIC S9(8) COMP.
      05 FILLER PIC X(36).

01 OWNER-DESCRIPTOR-TWO.
 03 OWNER-TWO-SET PIC X(16).
 03 OWNER-TWO-serial PIC S9(8) COMP.
 03 OWNER-TWO-KEY PIC X(40).
 03 OWNER-TWO-KEY-RDEF REDEFINES OWNER-TWO-KEY.
      05 OWNER-TWO-KEY-SERIAL PIC S9(8) COMP.
      05 FILLER PIC X(36).

01 OWNER-DESCRIPTOR-THREE.
 03 OWNER-THREE-SET PIC X(16).
 03 OWNER-THREE-serial PIC S9(8) COMP.
 03 OWNER-THREE-KEY PIC X(40).
 03 OWNER-THREE-KEY-RDEF REDEFINES OWNER-THREE-KEY.
      05 OWNER-THREE-KEY-SERIAL PIC S9(8) COMP.
      05 FILLER PIC X(36).

01 OCCURRENCE-DESCRIPTOR.
 03 RECORD-SR-NAME PIC X(18).
 03 FILLER PIC X(6) VALUE LOW-VALUES.
 03 RECORD-ID PIC S9(8) COMP.
 03 RECORD-SUGGESTED-PAGE PIC S9(8) COMP.
```
03 FILLER PIC S9(8) COMP VALUE ZERO.
03 RECORD-SERIAL PIC S9(8) COMP.
03 RECORD-LOAD-STATUS PIC X(4).
03 RECORD-DATA PIC X(2040).
03 FILLER REDEFINES RECORD-DATA.
05 RECORD-DATA-REDEF PIC X(40).
05 FILLER PIC X(2000).

* *

01 MISCELLANEOUS-FIELDS.
02 END-FLAG PIC XXX VALUE SPACES.
88 END-OF-DATA VALUE 'END'.
02 COUNTS.
03 SUM-CARDS-IN PIC 9(6) VALUE ZERO.
03 SUM-TRANSACTIONS PIC 9(6) VALUE ZERO.
02 ERROR-MESSAGE PIC X(30) VALUE SPACES.
02 I-CTRL PIC S9(4) COMP SYNC.
02 SAVE-COVERAGE-SERIAL PIC S9(8) COMP SYNC.

SKIP2

01 CARD-IMAGE.
02 CI-DATA-IMAGE.
03 CI-KEYFIELDS.
04 FILLER PIC X.
04 CI-CARD-TYPE PIC XX.
88 CI-END VALUE 'EN'.
04 CI-CARD-TYPE-RD REDEFINES CI-CARD-TYPE.
05 CI-CARD-TYPE-MAJ PIC X.
05 CI-CARD-TYPE-MIN PIC X.
88 CI-FIRST-PART VALUES ARE 'A', 'C', 'E', 'G', 'I', 'M', 'O', 'Q', 'S'.
04 FILLER PIC X.
04 CI-EMP-ID PIC 9(4).
04 CI-INSPLAN REDEFINES CI-EMP-ID.
05 CI-INSPLAN-CODE PIC 9(3).
05 FILLER PIC X.
04 CI-OFFICE REDEFINES CI-EMP-ID.
05 CI-OFFICE-CODE PIC 9(3).
05 FILLER PIC X.
03 CI-DATAFIELDS PIC X(72).

SKIP3

******************************************************************************************
* TRANSACTION-STORAGE:                                          *
* ONE CARD TYPE FOR EACH INPUT RECORD TYPE;                       *
* EACH CARD CONTAINS A CARD-TYPE CODE.                           *
* INPUT CARDS MAY INCLUDE KEYFIELDS USED BY                       *
* THE PROGRAM TO MAKE THE APPROPRIATE OWNER                      *
* RECORDS CURRENT BEFORE A MEMBER IS STORED.                     *
* TRANSACTION-STORAGE IS REDEFINED FOR THE                       *
* FORMAT OF EACH TYPE OF INPUT CARD.                             *
******************************************************************************************

01 TRANSACTION-STORAGE-AREA.
02 TRANSACTION-STORAGE-ALL.
03 TSA-SINGLE-CARD.
04 TSA-KEYFIELDS.
05 FILLER PIC X.
05 TSA-CARD-TYPE PIC XX.
88 TSA-DEPARTMENT VALUE IS 'D'.
88 TSA-EMPLOYEE VALUE IS 'E1'.
88 TSA-JOB VALUE IS 'J1'.
88 TSA-EMPOSITION VALUE IS 'P'.
88 TSA-EXPERTISE VALUE IS 'T'.
88 TSA-SKILL VALUE IS 'S'.
88 TSA-OFFICE VALUE IS 'O1'.
88 TSA-STRUCTURE VALUE IS '01'.
88 TSA-INS-PLAN-CODE VALUE IS 'I1'.
88 TSA-COVERAGE VALUE IS 'C'.
88 TSA-DENTAL VALUE IS 'L1'.
CA IDMS Reference - 19.0

88 TSA-HOSPITAL VALUE IS 'H1'.
88 TSA-NON-HOSP-CLAIM VALUE IS 'N1'.
04 FILLER PIC X(77).
03 TSA-OTHER-CARD-SPACE PIC X(400).
02 DEPT-STORAGE-AREA REDEFINES TRANSACTION-STORAGE-ALL.
03 D-CARD.
 04 D-KEYFIELDS.
   05 FILLER PIC X(4).
   05 D-DEPT-ID PIC 9(4).
 04 D-DATAFIELDS.
   05 D-DEPT-NAME PIC X(25).
   05 D-DEPT-HEAD-ID PIC 9(4).
   05 FILLER PIC X(40).
03 FILLER PIC X(400).
02 EMPLOYEE-STORAGE-AREA REDEFINES TRANSACTION-STORAGE-ALL.
03 E1-CARD.
 04 E1-KEYFIELDS.
   05 FILLER PIC X(4).
   05 E1-EMP-ID PIC 9(4).
 04 E1-DATAFIELDS.
   05 E1-EMP-NAME PIC X(25).
   05 E1-EMP-DEPT-ID PIC 9(4).
   05 E1-EMP-OFFICE PIC 9(3).
   05 FILLER PIC X(40).
03 E2-CARD.
 04 E2-KEYFIELDS.
   05 FILLER PIC X(4).
   05 E2-EMP-ID PIC 9(4).
 04 E2-DATAFIELDS.
   05 E2-EMP-ADDRESS PIC X(46).
   05 E2-EMP-PHONE PIC 9(10).
   05 E2-EMP-STATUS PIC 9(2).
   05 E2-EMP-SS-NUMBER PIC 9(9).
   05 FILLER PIC X(5).
03 E3-CARD.
 04 E3-KEYFIELDS.
   05 FILLER PIC X(4).
   05 E3-EMP-ID PIC 9(4).
 04 E3-DATAFIELDS.
   05 E3-EMP-START PIC 9(8).
   05 E3-EMP-DOB PIC 9(8).
   05 E3-EMP-TERM PIC 9(8).
   05 FILLER PIC X(48).
03 FILLER PIC X(240).
02 JOB-STORAGE-AREA REDEFINES TRANSACTION-STORAGE-ALL.
03 J1-CARD.
 04 J1-KEYFIELDS.
   05 FILLER PIC X(4).
   05 J1-JOB-ID PIC 9(4).
 04 J1-DATAFIELDS.
   05 J1-JOB-TITLE PIC X(20).
   05 J1-JOB-MIN-SAL PIC 9(8).
   05 J1-JOB-MAX-SAL PIC 9(8).
   05 J1-JOB-SAL-GRDS PIC 9(2) OCCURS 4.
   05 J1-JOB-NUM-POSTS PIC 9(3).
   05 J1-JOB-NUM-OPEN PIC 9(3).
   05 FILLER PIC X(22).
03 J2-CARD.
 04 J2-KEYFIELDS.
   05 FILLER PIC X(4).
 04 J2-DATAFIELDS.
   05 J2-JOB-DES-LINE PIC X(60).
   05 FILLER PIC X(16).
03 J3-CARD.
 04 J3-KEYFIELDS.
   05 FILLER PIC X(4).
 04 J3-DATAFIELDS.
   05 J3-JOB-DES-LINE PIC X(60).
05 FILLER PIC X(16).
03 J4-CARD.
  04 J4-KEYFIELDS.
    05 FILLER PIC X(4).
    04 J4-DATAFIELDS.
      05 J4-JOB-REQ-LINE PIC X(60).
      05 FILLER PIC X(16).
  03 J5-CARD.
    04 J5-KEYFIELDS.
      05 FILLER PIC X(4).
      04 J5-DATAFIELDS.
        05 J5-JOB-REQ-LINE PIC X(60).
        05 FILLER PIC X(80).
02 POSITION-STORAGE-AREA REdefines TRANSACTION-STORAGE-ALL.
  03 P-CARD.
    04 P-KEYFIELDS.
      05 FILLER PIC X(4).
      05 P-JOB-ID PIC 9(4).
      05 P-EMP-ID PIC 9(4).
    04 P-DATAFIELDS.
      05 P-START-DATE PIC 9(8).
      05 P-FINISH-DATE PIC 9(8).
      05 P-SALARY-GRADE PIC 9(2).
      05 P-SALARY-AMOUNT PIC 9(6)V99.
      05 P-BONUS-PERCENT PIC V999.
      05 P-COMM-PERCENT PIC V999.
      05 P-OVERTIME-RATE PIC 9V99.
      04 FILLER PIC X(33).
  03 FILLER PIC X(400).
02 EXPERTISE-STORAGE-AREA REdefines TRANSACTION-STORAGE-ALL.
  03 T-CARD.
    04 T-KEYFIELDS.
      05 FILLER PIC X(4).
      05 T-SKILL-ID PIC 9(4).
      05 T-EMP-ID PIC 9(4).
    04 T-DATAFIELDS.
      05 T-SKILL-LEVEL PIC 9(2).
      05 T-EXPERTISE-DATE PIC 9(8).
      05 FILLER PIC X(58).
  03 FILLER PIC X(400).
02 SKILL-STORAGE-AREA REdefines TRANSACTION-STORAGE-ALL.
  03 S-CARD.
    04 S-KEYFIELDS.
      05 FILLER PIC X(4).
      05 S-SKILL-ID PIC 9(4).
    04 S-DATAFIELDS.
      05 S-SKILL-NAME PIC X(12).
      05 S-SKILL-DESC PIC X(60).
  03 FILLER PIC X(400).
02 OFFICE-STORAGE-AREA REdefines TRANSACTION-STORAGE-ALL.
  03 01-CARD.
    04 01-KEYFIELDS.
      05 FILLER PIC X(4).
      05 01-OFFICE-CODE PIC 9(3).
    04 01-DATAFIELDS.
      05 01-OFFICE-ADDRESS PIC X(56).
      05 FILLER PIC X(17).
  03 02-CARD.
    04 02-KEYFIELDS.
      05 FILLER PIC X(4).
      05 02-OFFICE-CODE PIC 9(3).
    04 02-DATAFIELDS.
      05 02-OFFICE-PHONE PIC 9(7) OCCurs 3 TIMES.
      05 02-OFFICE-AREA PIC 9(3).
      05 02-OFFICE-SPEED-DIAL PIC 9(3).
      05 FILLER PIC X(46).
<table>
<thead>
<tr>
<th>03</th>
<th>FILLER</th>
<th>PIC X(320).</th>
</tr>
</thead>
<tbody>
<tr>
<td>02</td>
<td>STRUCTURE-STORAGE-AREA</td>
<td>REDEFINES TRANSACTION-STORAGE-ALL.</td>
</tr>
<tr>
<td>03</td>
<td>OG-CARD.</td>
<td></td>
</tr>
<tr>
<td>04</td>
<td>OG-KEYFIELDS.</td>
<td></td>
</tr>
<tr>
<td>05</td>
<td>FILLER</td>
<td>PIC X(4).</td>
</tr>
<tr>
<td>05</td>
<td>OG-EMP-RPTS-TO</td>
<td>PIC 9(4).</td>
</tr>
<tr>
<td>05</td>
<td>OG-EMP-MANAGES</td>
<td>PIC 9(4).</td>
</tr>
<tr>
<td>04</td>
<td>OG-DATAFIELDS.</td>
<td></td>
</tr>
<tr>
<td>05</td>
<td>OG-STRUCT-CODE</td>
<td>PIC X(2).</td>
</tr>
<tr>
<td>05</td>
<td>OG-RELATION-DATE</td>
<td>PIC 9(8).</td>
</tr>
<tr>
<td>05</td>
<td>FILLER</td>
<td>PIC X(58).</td>
</tr>
<tr>
<td>03</td>
<td>FILLER</td>
<td>PIC X(400).</td>
</tr>
<tr>
<td>02</td>
<td>INSURANCE-STORAGE-AREA</td>
<td>REDEFINES TRANSACTION-STORAGE-ALL.</td>
</tr>
<tr>
<td>03</td>
<td>I1-CARD.</td>
<td></td>
</tr>
<tr>
<td>04</td>
<td>I1-KEYFIELDS.</td>
<td></td>
</tr>
<tr>
<td>05</td>
<td>FILLER</td>
<td>PIC X(4).</td>
</tr>
<tr>
<td>05</td>
<td>I1-INSPLAN-CODE</td>
<td>PIC X(3).</td>
</tr>
<tr>
<td>04</td>
<td>I1-DATAFIELDS.</td>
<td></td>
</tr>
<tr>
<td>05</td>
<td>I1-INSPLAN-CO-NAME</td>
<td>PIC X(45).</td>
</tr>
<tr>
<td>05</td>
<td>FILLER</td>
<td>PIC X(28).</td>
</tr>
<tr>
<td>03</td>
<td>I2-CARD.</td>
<td></td>
</tr>
<tr>
<td>04</td>
<td>I2-KEYFIELDS.</td>
<td></td>
</tr>
<tr>
<td>05</td>
<td>FILLER</td>
<td>PIC X(4).</td>
</tr>
<tr>
<td>05</td>
<td>I2-INSPLAN-CODE</td>
<td>PIC X(3).</td>
</tr>
<tr>
<td>04</td>
<td>I2-DATAFIELDS.</td>
<td></td>
</tr>
<tr>
<td>05</td>
<td>I2-CO-ADDRESS</td>
<td>PIC X(46).</td>
</tr>
<tr>
<td>05</td>
<td>I2-CO-PHONE</td>
<td>PIC 9(10).</td>
</tr>
<tr>
<td>05</td>
<td>FILLER</td>
<td>PIC X(17).</td>
</tr>
<tr>
<td>03</td>
<td>I3-CARD.</td>
<td></td>
</tr>
<tr>
<td>04</td>
<td>I3-KEYFIELDS.</td>
<td></td>
</tr>
<tr>
<td>05</td>
<td>FILLER</td>
<td>PIC X(4).</td>
</tr>
<tr>
<td>05</td>
<td>I3-INSPLAN-CODE</td>
<td>PIC X(3).</td>
</tr>
<tr>
<td>04</td>
<td>I3-DATAFIELDS.</td>
<td></td>
</tr>
<tr>
<td>05</td>
<td>I3-GROUP-NUM</td>
<td>PIC 9(6).</td>
</tr>
<tr>
<td>05</td>
<td>I3-DESCRIPTION.</td>
<td></td>
</tr>
<tr>
<td>06</td>
<td>I3-DEDUCT</td>
<td>PIC 9(6)V99.</td>
</tr>
<tr>
<td>06</td>
<td>I3-MAX-LIFE-COST</td>
<td>PIC 9(6)V99.</td>
</tr>
<tr>
<td>06</td>
<td>I3-FAM-COST</td>
<td>PIC 9(6)V99.</td>
</tr>
<tr>
<td>06</td>
<td>I3-DEP-COST</td>
<td>PIC 9(6)V99.</td>
</tr>
<tr>
<td>05</td>
<td>FILLER</td>
<td>PIC X(35).</td>
</tr>
<tr>
<td>03</td>
<td>FILLER</td>
<td>PIC X(240).</td>
</tr>
<tr>
<td>02</td>
<td>COVERAGE-STORAGE-AREA</td>
<td>REDEFINES TRANSACTION-STORAGE-ALL.</td>
</tr>
<tr>
<td>03</td>
<td>C-CARD.</td>
<td></td>
</tr>
<tr>
<td>04</td>
<td>C-KEYFIELDS.</td>
<td></td>
</tr>
<tr>
<td>05</td>
<td>FILLER</td>
<td>PIC X(4).</td>
</tr>
<tr>
<td>05</td>
<td>C-INSPLAN-CODE</td>
<td>PIC X(3).</td>
</tr>
<tr>
<td>05</td>
<td>C-EMP-ID</td>
<td>PIC 9(4).</td>
</tr>
<tr>
<td>04</td>
<td>C-DATAFIELDS.</td>
<td></td>
</tr>
<tr>
<td>05</td>
<td>C-SELECT-DATE</td>
<td>PIC 9(8).</td>
</tr>
<tr>
<td>05</td>
<td>C-TERMIN-DATE</td>
<td>PIC 9(8).</td>
</tr>
<tr>
<td>05</td>
<td>C-TYPE</td>
<td>PIC X.</td>
</tr>
<tr>
<td>05</td>
<td>C-INSPLAN-CODE</td>
<td>PIC X(3).</td>
</tr>
<tr>
<td>05</td>
<td>FILLER</td>
<td>PIC X(49).</td>
</tr>
<tr>
<td>03</td>
<td>FILLER</td>
<td>PIC X(400).</td>
</tr>
<tr>
<td>02</td>
<td>DENTAL-STORAGE-AREA</td>
<td>REDEFINES TRANSACTION-STORAGE-ALL.</td>
</tr>
<tr>
<td>03</td>
<td>L1-CARD.</td>
<td></td>
</tr>
<tr>
<td>04</td>
<td>L1-KEYFIELDS.</td>
<td></td>
</tr>
<tr>
<td>05</td>
<td>FILLER</td>
<td>PIC X(4).</td>
</tr>
<tr>
<td>04</td>
<td>L1-DATAFIELDS.</td>
<td></td>
</tr>
<tr>
<td>05</td>
<td>L1-DC-CLAIM-DATE</td>
<td>PIC 9(8).</td>
</tr>
<tr>
<td>05</td>
<td>L1-DC-PATIENT-NAME</td>
<td>PIC X(25).</td>
</tr>
<tr>
<td>05</td>
<td>L1-DC-PATIENT-DOB</td>
<td>PIC 9(8).</td>
</tr>
<tr>
<td>05</td>
<td>L1-DC-SEX</td>
<td>PIC X.</td>
</tr>
<tr>
<td>05</td>
<td>L1-DC-REL-TO-EMP</td>
<td>PIC X(10).</td>
</tr>
<tr>
<td>05</td>
<td>L1-DC-DENTIST-NAME</td>
<td>PIC X(24).</td>
</tr>
<tr>
<td>03</td>
<td>L2-CARD.</td>
<td></td>
</tr>
<tr>
<td>04</td>
<td>L2-KEYFIELDS.</td>
<td></td>
</tr>
</tbody>
</table>
05 FILLER PIC X(4).
04 L2-DATAFIELDS.
05 L2-DC-DENTIST-ADDRESS PIC X(46).
05 L2-DC-DENTIST-LIC-NUM PIC 9(6).
05 L2-DC-NUM-PROCEDURES PIC 9(2).
05 FILLER PIC X(22).
03 LA-CARD.
04 LA-KEYFIELDS.
05 FILLER PIC X(4).
04 LA-DATAFIELDS.
05 LA-DC-TOOTH-NUM PIC 9(2).
05 LA-DC-SERVICE-DATE PIC 9(8).
05 LA-DC-PROC-CODE PIC 9(4).
05 LA-DC-FEE PIC 9(6)V99.
05 FILLER PIC X(54).
03 LB-CARD.
04 LB-KEYFIELDS.
05 FILLER PIC X(4).
04 LB-DATAFIELDS.
05 LB-DC-DESC-OF-SERVICE PIC X(60).
05 FILLER PIC X(16).
03 FILLER PIC X(160).
02 HOSPITAL-STORAGE-AREA REDEFINES TRANSACTION-STORAGE-ALL.
03 H1-CARD.
04 H1-KEYFIELDS.
05 FILLER PIC X(4).
04 H1-DATAFIELDS.
05 H1-HC-CLAIM-DATE PIC 9(8).
05 H1-HC-PATIENT-NAME PIC X(25).
05 H1-HC-PATIENT-DOB PIC 9(8).
05 H1-HC-SEX PIC X.
05 H1-HC-REL-TO-EMP PIC X(10).
05 H1-HC-HOSP-NAME PIC X(24).
03 H2-CARD.
04 H2-KEYFIELDS.
05 FILLER PIC X(4).
04 H2-DATAFIELDS.
05 H2-HC-HOSP-ADDRESS PIC X(46).
05 H2-HC-ADMIT-DATE PIC 9(8).
05 H2-HC-DISCH-DATE PIC 9(8).
05 FILLER PIC X(14).
03 H3-CARD.
04 H3-KEYFIELDS.
05 FILLER PIC X(4).
04 H3-DATAFIELDS.
05 H3-HC-DIAGNOSIS PIC X(60).
05 FILLER PIC X(16).
03 H4-CARD.
04 H4-KEYFIELDS.
05 FILLER PIC X(4).
04 H4-DATAFIELDS.
05 H4-HC-DIAGNOSIS PIC X(60).
05 FILLER PIC X(16).
03 H5-CARD.
04 H5-KEYFIELDS.
05 FILLER PIC X(4).
04 H5-DATAFIELDS.
05 H5-HOSP-CHARGES.
06 H5-HC-WARD.
07 H5-HC-WARD-DAYS PIC 9(4).
07 H5-HC-WARD-RATE PIC 9(6)V99.
07 H5-HC-WARD-TOTAL PIC 9(6)V99.
06 H5-SEMI-PRIVATE.
07 H5-HC-SEMI-DAYS PIC 9(4).
07 H5-HC-SEMI-RATE PIC 9(6)V99.
07 H5-HC-SEMI-TOTAL PIC 9(6)V99.
06 H5-SEMI-OTHER.
07 H5-HC-DEL-COST PIC 9(6)V99.
07 H5-HC-ANESTH-COST PIC 9(6)V99.
07 H5-HC-LAB-COST PIC 9(6)V99.
05 FILLER PIC X(12).
03 FILLER PIC X(80).
02 NONHOSP-DOCKET-AREA REDEFINES TRANSACTION-DOCKET-ALL.

03 N1-CARD.
  04 N1-KEYFIELDS.
    05 FILLER PIC X(4).
  04 N1-DATAFIELDS.
    05 N1-NC-CLAIM-DATE PIC 9(8).
    05 N1-NC-PATIENT-NAME PIC X(25).
    05 N1-NC-PATIENT-DOB PIC 9(8).
    05 N1-NC-SEX PIC X.
    05 N1-NC-REL-TO-EMP PIC X(10).
    05 N1-NC-PHYS-NAME PIC X(24).

03 N2-CARD.
  04 N2-KEYFIELDS.
    05 FILLER PIC X(4).
  04 N2-DATAFIELDS.
    05 N2-NC-PHYS-ADDRESS PIC X(46).
    05 N2-NC-PHYS-ID PIC 9(6).
    05 N2-NC-NUM-PROCS PIC 9(2).
    05 FILLER PIC X(22).

03 N3-CARD.
  04 N3-KEYFIELDS.
    05 FILLER PIC X(4).
  04 N3-DATAFIELDS.
    05 N3-NC-DIAGNOSIS PIC X(60).
    05 FILLER PIC X(16).

03 N4-CARD.
  04 N4-KEYFIELDS.
    05 FILLER PIC X(4).
  04 N4-DATAFIELDS.
    05 N4-NC-DIAGNOSIS PIC X(60).
    05 FILLER PIC X(16).

03 NA-CARD.
  04 NA-KEYFIELDS.
    05 FILLER PIC X(4).
  04 NA-DATAFIELDS.
    05 NA-NC-SERVICE-DATE PIC 9(8).
    05 NA-NC-PROC-CODE PIC 9(4).
    05 NA-NC-FEE PIC 9(6)V99.
    05 FILLER PIC X(56).

03 NB-CARD.
  04 NB-KEYFIELDS.
    05 FILLER PIC X(4).
  04 NB-DATAFIELDS.
    05 NB-NC-DESC-OF-SERVICE PIC X(62).
    05 FILLER PIC X(14).

01 NAMES-INFO.
  02 NAMES-SSNAME PIC X(8) VALUE 'EMPSS01'.
  02 NAMES-DBNAME PIC X(8) VALUE 'EMPDEMO'.
  02 NAMES-DMCLNAME PIC X(8) VALUE 'IDMSDMCL'.

PROCEDURE DIVISION.

**********************************************************************
* PROCEDURE DIVISION GENERAL STRATEGY:                             *
* 1) READ 1 OR MORE CARDS TO FOR A TRANSACTION                      *
* 2) PERFORM THE APPROPRIATE ROUTINE, BASED UPON THE                *
* TRANSACTION CODE                                                 *
* 3) CONTINUE UNTIL ALL CARD INPUT IS EXHAUSTED                      *
**********************************************************************

0000-MAIN-LINE SECTION.
0001-SETUP.
  DISPLAY '*** BEFORE FIRST CALL ***'.
  CALL 'IDMSDBLU' USING NAMES-INFO.

0005-ML-START.
ACCEPT CARD-IMAGE.
DISPLAY '*** AFTER ACCEPT ***'
PERFORM 0020-MAIN-LOOP THRU 0020-ML-EXIT UNTIL END-OF-DATA.
PERFORM 9999-END.
0020-MAIN-LOOP.
PERFORM 0510-READ-TRANSACTION THRU 0515-RT-EXIT.
DISPLAY '*** AFTER PERFORM 510- ***'.
IF END-OF-DATA
GO TO 0020-ML-EXIT.
ADD 1 TO SUM-TRANSACTIONS.
IF TSA-DEPARTMENT
PERFORM 1010-DO-DEPARTMENT THRU 1090-DD-EXIT ELSE
IF TSA-EMPLOYEE
PERFORM 1510-DO-EMPLOYEE THRU 1590-DE-EXIT ELSE
IF TSA-JOB
PERFORM 2010-DO-JOB THRU 2090-DJ-EXIT ELSE
IF TSA-EMPOSITION
PERFORM 2510-DO-EMPOSITION THRU 2590-DEM-EXIT ELSE
IF TSA-EXPERTISE
PERFORM 3010-DO-EXPERTISE THRU 3090-DEX-EXIT ELSE
IF TSA-SKILL
PERFORM 3510-DO-SKILL THRU 3590-DS-EXIT ELSE
IF TSA-OFFICE
PERFORM 4510-DO-OFFICE THRU 4590-DO-EXIT ELSE
IF TSA-STRUCTURE
PERFORM 5010-DO-STRUCTURE THRU 5090-DS-EXIT ELSE
IF TSA-INS-PLAN-CODE
PERFORM 5510-DO-INSURANCE THRU 5590-DI-EXIT ELSE
IF TSA-COVERAGE
PERFORM 6010-DO-COVERAGE THRU 6090-DC-EXIT ELSE
IF TSA-DENTAL
PERFORM 6510-DO-DENTAL THRU 6590-DDN-EXIT ELSE
IF TSA-HOSPITAL
PERFORM 7010-DO-HOSPITAL THRU 7090-DH-EXIT ELSE
PERFORM 7510-DO-NONHOSP THRU 7590-DN-EXIT.
0020-ML-EXIT.
EXIT.
**************************************
*      UTILITY ROUTINES FOLLOW       *
**************************************
0500-UTILITY SECTION.
*****************************************************************************
*    THIS ROUTINE ASSEMBLES A TRANSACTION FROM ONE OR MORE                *
*    INDIVIDUAL CARDS;  NOTE THAT WHEN THIS PROCEDURE IS                *
*    ENTERED, A CARD IS ALWAYS PRESENT IN THE 'CARD IMAGE'                *
*    BUFFER.                                                              *
*                                                                        *
*    DEPARTMENT HAS A SINGLE 'D' CARD                                    *
*    EMPLOYEE HAS AN 'E1', AN 'E2', AND AN 'E3' CARD                     *
*    JOB HAS 'J1' THRU 'J5' CARDS                                       *
*    EMPOSITION HAS A SINGLE 'P ' CARD                                   *
*    EXPERTISE HAS A SINGLE 'T ' CARD                                   *
*    SKILL HAS A SINGLE 'S ' CARD                                       *
*    OFFICE HAS AN 'O1' AND AN 'O2' CARD                                 *
* STRUCTURE HAS A SINGLE 'OG' CARD *
* INSURANCE-PLAN HAS AN 'I1', AN 'I2', AND AN 'I3' CARD *
* COVERAGE HAS A SINGLE 'C' CARD *
* DENTAL-CLAIM HAS AN 'L1' AND AN 'L2' CARD, FOLLOWED *
  * BY 2 TO 20 'LX' CARDS (WHERE 'X' IS A LETTER *
  * FROM A TO T) *
* HOSPITAL-CLAIM HAS 'H1' THRU 'H5' CARDS *
* NON-HOSP-CLAIM HAS 'N1' THRU 'N4' CARDS, FOLLOWED *
  * BY 2 TO 20 'NX' CARDS (WHERE 'X' IS A LETTER *
  * FROM A TO T) *

***********************************************************************
0510-READ-TRANSACTION.
MOVE SPACES TO TRANSACTION-STORAGE-AREA.

IF CI-END
  MOVE 'END' TO END-FLAG
  GO TO 0515-RT-EXIT.

IF CI-CARD-TYPE = 'D' OR 'P' OR 'T' OR 'S' OR 'OG' OR 'C'
  MOVE CI-DATA-IMAGE TO TSA-SINGLE-CARD
  PERFORM 0600-READ-CARD THRU 0615-RC-EXIT
ELSE
IF CI-CARD-TYPE = 'E1'
  PERFORM 0520-ASSEM-EMPLOYEE THRU 0528-AE-EXIT
ELSE
IF CI-CARD-TYPE = 'J1'
  PERFORM 0540-ASSEM-JOB THRU 0548-AJ-EXIT
ELSE
IF CI-CARD-TYPE = 'I1'
  PERFORM 0550-ASSEM-INS THRU 0558-AI-EXIT
ELSE
IF CI-CARD-TYPE = 'L1'
  PERFORM 0560-ASSEM-DENT THRU 0568-AD-EXIT
ELSE
IF CI-CARD-TYPE = 'H1'
  PERFORM 0570-ASSEM-HOSP THRU 0578-AH-EXIT
ELSE
  MOVE 'INVALID CARD TYPE/SEQ' TO ERROR-MESSAGE
  PERFORM 0620-DISPLAY-CARD-ERROR THRU 0640-DCE-EXIT
  PERFORM 0600-READ-CARD THRU 0615-RC-EXIT
  GO TO 0510-READ-TRANSACTION.

0515-RT-EXIT.
EXIT.

***********************************************************************
* THE FOLLOWING MODULES ASSEMBLE MULTIPLE INPUT CARDS *
* INTO THE APPROPRIATE WORK RECORDS. *
***********************************************************************

0520-ASSEM-EMPLOYEE.
MOVE CI-DATA-IMAGE TO E1-CARD.

PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

IF CI-CARD-TYPE = 'E2'
  AND CI-EMP-ID = E1-EMP-ID
  MOVE CI-DATA-IMAGE TO E2-CARD
  PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.
IF CI-CARD-TYPE = 'E3'
  AND CI-EMP-ID = E1-EMP-ID
  MOVE CI-DATA-IMAGE TO E3-CARD
  PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.
0528-AE-EXIT.
  EXIT.

0530-ASSEM-OFFICE.
  MOVE CI-DATA-IMAGE  TO 01-CARD.
  PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.
  IF CI-CARD-TYPE = '02'
    AND CI-OFFICE-CODE = 01-OFFICE-CODE
    MOVE CI-DATA-IMAGE  TO 02-CARD
    PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

0530-AO-EXIT.
  EXIT.

0540-ASSEM-JOB.
  MOVE CI-DATA-IMAGE  TO J1-CARD.
  PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.
  IF CI-CARD-TYPE = 'J2'
    MOVE CI-DATA-IMAGE  TO J2-CARD
    PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.
  IF CI-CARD-TYPE = 'J3'
    MOVE CI-DATA-IMAGE  TO J3-CARD
    PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.
  IF CI-CARD-TYPE = 'J4'
    MOVE CI-DATA-IMAGE  TO J4-CARD
    PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.
  IF CI-CARD-TYPE = 'J5'
    MOVE CI-DATA-IMAGE  TO J5-CARD
    PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

0548-AJ-EXIT.
  EXIT.

0550-ASSEM-INS.
  MOVE CI-DATA-IMAGE  TO I1-CARD.
  PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.
  IF CI-CARD-TYPE = 'I2'
    AND CI-INSPLAN-CODE = I1-INSPLAN-CODE
    MOVE CI-DATA-IMAGE  TO I2-CARD
    PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.
  IF CI-CARD-TYPE = 'I3'
    AND CI-INSPLAN-CODE = I1-INSPLAN-CODE
    MOVE CI-DATA-IMAGE  TO I3-CARD
    PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

0550-AI-EXIT.
  EXIT.

0560-ASSEM-DENT.
  MOVE CI-DATA-IMAGE TO L1-CARD.
  PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.
  IF CI-CARD-TYPE = 'L2'
    MOVE CI-DATA-IMAGE  TO L2-CARD
    PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.
  MOVE 0 TO I-CTRL.
  PERFORM 0563-GET-CHARGES THRU 0563-GC-EXIT
  UNTIL (CI-CARD-TYPE-MAJ NOT = 'L' OR
         CI-CARD-TYPE-MIN NOT ALPHABETIC).

0568-AD-EXIT.
  EXIT.

0563-GET-CHARGES.
  IF CI-FIRST-PART
    ADD 1 TO I-CTRL
MOVE CI-DATA-IMAGE TO LA-CARD
MOVE LA-DC-TOOTH-NUM TO TOOTH-NUMBER-0405 (I-CTRL)
MOVE LA-DC-SERVICE-DATE TO SERVICE-DATE-0405 (I-CTRL)
MOVE LA-DC-PROC-CODE TO PROCEDURE-CODE-0405 (I-CTRL)
MOVE LA-DC-FEE TO FEE-0405 (I-CTRL).

IF CI-2ND-PART
  MOVE CI-DATA-IMAGE TO LB-CARD
  MOVE LB-DC-DESC-OF-SERVICE TO DESCRIPTION-OF-SERVICE-0405 (I-CTRL).
END-IF

PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.
0563-GC-EXIT.
EXIT.
0570-ASSEM-HOSP.
  MOVE CI-DATA-IMAGE TO H1-CARD.
  PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.
  IF CI-CARD-TYPE = 'H2'
    MOVE CI-DATA-IMAGE TO H2-CARD
    PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.
  END-IF
  IF CI-CARD-TYPE = 'H3'
    MOVE CI-DATA-IMAGE TO H3-CARD
    PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.
  END-IF
  IF CI-CARD-TYPE = 'H4'
    MOVE CI-DATA-IMAGE TO H4-CARD
    PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.
  END-IF
  IF CI-CARD-TYPE = 'H5'
    MOVE CI-DATA-IMAGE TO H5-CARD
    PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.
  END-IF
0578-AH-EXIT.
EXIT.
0580-ASSEM-NONHOSP.
  MOVE CI-DATA-IMAGE TO N1-CARD.
  PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.
  IF CI-CARD-TYPE = 'N2'
    MOVE CI-DATA-IMAGE TO N2-CARD
    PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.
  END-IF
  IF CI-CARD-TYPE = 'N3'
    MOVE CI-DATA-IMAGE TO N3-CARD
    PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.
  END-IF
  IF CI-CARD-TYPE = 'N4'
    MOVE CI-DATA-IMAGE TO N4-CARD
    PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.
  END-IF
  MOVE 0 TO I-CTRL.
  PERFORM 0583-GET-CHARGES THRU 0583-GC-EXIT
  UNTIL (CI-CARD-TYPE-MAJ NOT = 'N' OR CI-CARD-TYPE-MIN NOT ALPHABETIC).
0588-AN-EXIT.
EXIT.
0583-GET-CHARGES.
  IF CI-FIRST-PART
    ADD 1 TO I-CTRL
    MOVE CI-DATA-IMAGE TO NA-CARD
    MOVE NA-NC-SERVICE-DATE TO SERVICE-DATE-0445 (I-CTRL)
    MOVE NA-NC-PROC-CODE TO PROCEDURE-CODE-0445 (I-CTRL)
    MOVE NA-NC-FEE TO FEE-0445 (I-CTRL).
  END-IF
  IF CI-2ND-PART
    MOVE CI-DATA-IMAGE TO NB-CARD
MOVE NB-NC-DESC-OF-SERVICE TO DESCRIPTION-OF-SERVICE-0445 (I-CTRL).

PERFORM 0600-READ-CARD THRU 0615-RC-EXIT.

0583-GC-EXIT.
EXIT.

0600-READ-CARD.
IF CARD-COUNT = '50'
  MOVE ZERO TO CARD-COUNT.
  DISPLAY CARD-IMAGE.
  ACCEPT CARD-IMAGE.
  ADD 1 TO CARD-COUNT.
  ADD 1 TO SUM-CARDS-IN.

0615-RC-EXIT.
EXIT.

0620-DISPLAY-CARD-ERROR.
DISPLAY ERROR-MESSAGE, CARD-IMAGE.

0640-DCE-EXIT.
EXIT.

1000-PROCESS SECTION.
*
***************************************************************
*   THIS MAIN PROCESS SECTION HANDLES ALL FORMATTING OF     *
*   OWNER DESCRIPTOR RECORDS AND CALLS TO IDMSDBLU.          *
***************************************************************

1010-DO-DEPARTMENT.
****************************************************
*   THIS ROUTINE STORES A DEPARTMENT RECORD.             *
****************************************************

***BUILD RECORD OCCURRENCE OF DEPARTMENT RECORD**********
  MOVE D-DEPT-ID TO DEPT-ID-0410.
  MOVE D-DEPT-NAME TO DEPT-NAME-0410.
  MOVE D-DEPT-HEAD-ID TO DEPT-HEAD-ID-0410.

1020-STORE-DEPT.
  MOVE 'DEPARTMENT' TO RECORD-SR-NAME.
  MOVE 410 TO RECORD-ID.
  MOVE DEPARTMENT TO RECORD-DATA.
  CALL 'IDMSDBLU' USING OCCURRENCE-DESCRIPTOR.
  PERFORM DBLU-STATUS.

1090-DD-EXIT.
EXIT.

1510-DO-EMPLOYEE.
******************************************************************************
*   THIS ROUTINE STORES THE EMPLOYEE RECORD. THE OWNERS IN *  
*   THE DEPT-EMPLOYEE AND OFFICE-EMPLOYEE SETS MUST BE     *
*   PRESENT BY THE END OF THE RUN.                         *
******************************************************************************

*****BUILD OWNER OCCURRENCE OF SKILL-NAME-NDX SET************
  MOVE 'EMP-NAME-NDX' TO OWNER-ONE-SET.
  MOVE -1 TO OWNER-ONE-SERIAL.

*****BUILD OWNER OCCURRENCE OF DEPT-EMPLOYEE SET************
  MOVE 'DEPT-EMPLOYEE' TO OWNER-TWO-SET.
  MOVE -1 TO OWNER-TWO-SERIAL.
  MOVE E1-EMP-DEPT-ID TO OWNER-TWO-KEY.
*****BUILD OWNER OCCURRENCE OF OFFICE-EMPLOYEE SET***************

MOVE 'OFFICE-EMPLOYEE' TO OWNER-THREE-SET.
MOVE -1 TO OWNER-THREE-SERIAL.
MOVE E1-EMP-OFFICE TO OWNER-THREE-KEY.

*****BUILD RECORD OCCURRENCE OF EMPLOYEE RECORD***************

MOVE E1-EMP-ID TO EMP-ID-0415.
MOVE E1-EMP-NAME TO EMP-NAME-0415.
MOVE E2-EMP-ADDRESS TO EMP-ADDRESS-0415.
MOVE E2-EMP-PHONE TO EMP-PHONE-0415.
MOVE E2-EMP-STATUS TO STATUS-0415.
IF STATUS-0415 EQUAL TO '05'
    MOVE ZEROS TO TERMINATION-DATE-0415
ELSE
    MOVE E3-EMP-TERM TO TERMINATION-DATE-0415.
MOVE E2-EMP-SS-NUMBER TO SS-NUMBER-0415.
MOVE E3-EMP-START TO START-DATE-0415.
MOVE E3-EMP-DOB TO BIRTH-DATE-0415.
1520-STORE-EMP.
MOVE 'EMPLOYEE' TO RECORD-SR-NAME.
MOVE 415 TO RECORD-ID.
MOVE EMPLOYEE TO RECORD-DATA.
CALL 'IDMSDBLU' USING OCCURRENCE-DESCRIPTOR
    OWNER-DESCRIPTOR-ONE
    OWNER-DESCRIPTOR-TWO
    OWNER-DESCRIPTOR-THREE.
PERFORM DBLU-STATUS.
1590-DE-EXIT.
EXIT.

2010-DO-JOB.

**********************************************
* THIS ROUTINE STORES THE JOB RECORD *
**********************************************

*****BUILD OWNER OCCURRENCE OF JOB-TITLE-NDX SET***************

MOVE 'JOB-TITLE-NDX' TO OWNER-ONE-SET.
MOVE -1 TO OWNER-ONE-SERIAL.
MOVE -1 TO OWNER-ONE-KEY-SERIAL.

*****BUILD RECORD OCCURRENCE OF JOB RECORD***************

MOVE J1-JOB-ID TO JOB-ID-0440.
MOVE J1-JOB-TITLE TO TITLE-0440.
MOVE J1-JOB-MIN-SAL TO MINIMUM-SALARY-0440.
MOVE J1-JOB-MAX-SAL TO MAXIMUM-SALARY-0440.
MOVE J1-JOB-NUM-POSTS TO NUMBER-OF-POSITIONS-0440.
MOVE J1-JOB-NUM-OPEN TO NUMBER-OPEN-0440.
MOVE J2-JOB-DES-LINE TO DESCRIPTION-LINE-0440 (1).
MOVE J3-JOB-DES-LINE TO DESCRIPTION-LINE-0440 (2).
MOVE J4-JOB-REQ-LINE TO REQUIREMENT-LINE-0440 (1).
MOVE J5-JOB-REQ-LINE TO REQUIREMENT-LINE-0440 (2).
MOVE 0 TO I-CTRL.
PERFORM 2110-DO-SAL-GRDS THRU 2110-DSG-EXIT
    VARYING I-CTRL FROM 1 BY 1 UNTIL I-CTRL = +4.

2020-STORE-JOB.
MOVE 'JOB' TO RECORD-SR-NAME.
MOVE 440 TO RECORD-ID.
MOVE JOB TO RECORD-DATA.
CALL 'IDMSDBLU' USING OCCURRENCE-DESCRIPTOR
    OWNER-DESCRIPTOR-ONE.
PERFORM DBLU-STATUS.
2090-DJ-EXIT.
EXIT.

2110-DO-SAL-GRDS.
MOVE J1-JOB-SAL-GRDS (I-CTRL)
TO SALARY-GRAD-0440 (I-CTRL).
2110-DSG-EXIT.
EXIT.
2510-DO-EMPOSITION.

**************************************************************
* THIS ROUTINE STORES THE EMPOSITION RECORD. THE OWNERS   *
* THE JOB-EMPOSITION AND EMP-EMPOSITION SETS MUST BE      *
* PRESENT BY THE END OF THE RUN.                         *
**************************************************************

*****BUILD OWNER OCCURRENCE FOR THE EMP-EMPOSITION SET*****
MOVE 'EMP-EMPOSITION' TO OWNER-TWO-SET.
MOVE -1 TO OWNER-TWO-SERIAL.
MOVE P-EMP-ID TO OWNER-TWO-KEY.

*****BUILD OWNER OCCURRENCE FOR THE JOB-EMPOSITION SET*****
MOVE 'JOB-EMPOSITION' TO OWNER-ONE-SET.
MOVE -1 TO OWNER-ONE-SERIAL.
MOVE P-JOB-ID TO OWNER-ONE-KEY.

*****BUILD RECORD OCCURRENCE OF EMPOSITION RECORD***********
MOVE P-START-DATE TO START-DATE-0420.
MOVE P-FINISH-DATE TO FINISH-DATE-0420.
MOVE P-SALARY-GRAD TO SALARY-GRAD-0420.
MOVE P-SALARY-AMOUNT TO SALARY-AMOUNT-0420.
MOVE P-BONUS-PERCENT TO BONUS-PERCENT-0420.
MOVE P-COMM-PERCENT TO COMMISSION-PERCENT-0420.
MOVE P-OVERTIME-RATE TO OVERTIME-RATE-0420.
2520-STORE-EMPOSITION.

MOVE EMPOSITION TO RECORD-SR-NAME.
MOVE 420 TO RECORD-ID.
MOVE EMPOSITION TO RECORD-DATA.
CALL 'IDMSDBLU' USING OCCURRENCE-DESCRIPTOR
OWNER-DESCRIPTOR-ONE
OWNER-DESCRIPTOR-TWO.

2590-DEM-EXIT.
EXIT.
3010-DO-EXPERTISE.

**************************************************************
* THE NEXT ROUTINE STORES A NEW EXPERTISE RECORD. THE       *
* SKILL AND EMPLOYEE OWNER RECORDS MUST BE PRESENT          *
* BY THE END OF THE RUN.                                    *
**************************************************************

*****BUILD OWNER OCCURRENCE FOR THE EMP-EXPERTISE SET*****
MOVE 'EMP-EXPERTISE' TO OWNER-TWO-SET.
MOVE -1 TO OWNER-TWO-SERIAL.
MOVE T-EMP-ID TO OWNER-TWO-KEY.

*****BUILD OWNER OCCURRENCE FOR SKILL-EXPERTISE SET******
MOVE 'SKILL-EXPERTISE' TO OWNER-ONE-SET.
MOVE -1 TO OWNER-ONE-SERIAL.
MOVE T-SKILL-ID TO OWNER-ONE-KEY.

*****BUILD OCCURRENCE OF EXPERTISE RECORD*******************
MOVE T-SKILL-LEVEL TO SKILL-LEVEL-0425.
MOVE T-EXPERTISE-DATE TO EXPERTISE-DATE-0425.
3020-STORE-EXPERTISE.

MOVE 'EXPERTISE' TO RECORD-SR-NAME.
MOVE 425 TO RECORD-ID.
MOVE EXPERTISE TO RECORD-DATA.
CALL 'IDMSDBLU' USING OCCURRENCE-DESCRIPTOR
OWNER-DESCRIPTOR-ONE
OWNER-DESCRIPTOR-TWO.
PERFORM DBLU-STATUS.

3090-DEX-EXIT.
EXIT.
3510-DO-SKILL.

***************************************************************
* THIS ROUTINE STORES A NEW SKILL RECORD.                      *
***************************************************************
*****BUILD OWNER OCCURRENCE FOR THE SKILL-NAME-NDX SET*******
    MOVE 'SKILL-NAME-NDX' TO OWNER-ONE-SET.
    MOVE -1 TO OWNER-ONE-SERIAL.
    MOVE -1 TO OWNER-ONE-KEY-SERIAL.

*****BUILD OCCURRENCE OF SKILL RECORD**************************
    MOVE S-SKILL-ID TO SKILL-ID-0455.
    MOVE S-SKILL-NAME TO SKILL-NAME-0455.
    MOVE S-SKILL-DESC TO SKILL-DESCRIPTION-0455.
3520-STORE-SKILL.
    MOVE 'SKILL' TO RECORD-SR-NAME.
    MOVE 455 TO RECORD-ID.
    MOVE SKILL TO RECORD-DATA.
    CALL 'IDMSDBLU' USING OCCURRENCE-DESCRIPTOR
    OWNER-DESCRIPTOR-ONE.
    PERFORM DBLU-STATUS.

3590-DS-EXIT.
EXIT.
4510-DO-OFFICE.

***************************************************************
* THIS ROUTINE STORES A NEW OFFICE RECORD                      *
***************************************************************
*****BUILD OCCURRENCE OF OFFICE RECORD*************************
    MOVE 01-OFFICE-CODE TO OFFICE-CODE-0450.
    MOVE 01-OFFICE-ADDRESS TO OFFICE-ADDRESS-0450.
    MOVE 02-OFFICE-AREA TO OFFICE-AREA-CODE-0450.
    MOVE 02-OFFICE-SPEED-DIAL TO SPEED-DIAL-0450.
    PERFORM 4615-OFFICE-PHONE THRU 4615-OP-EXIT
    VARYING I-CTRL FROM 1 BY 1 UNTIL I-CTRL = +4.
4520-STORE-OFFICE.
    MOVE 'OFFICE' TO RECORD-SR-NAME.
    MOVE 450 TO RECORD-ID.
    MOVE OFFICE TO RECORD-DATA.
    CALL 'IDMSDBLU' USING OCCURRENCE-DESCRIPTOR.
    PERFORM DBLU-STATUS.

4590-DO-EXIT.
EXIT.
4615-OFFICE-PHONE.
    IF 02-OFFICE-PHONE (I-CTRL) IS NOT NUMERIC
    MOVE ZEROS TO OFFICE-PHONE-0450 (I-CTRL)
    ELSE
    MOVE 02-OFFICE-PHONE (I-CTRL)
    TO OFFICE-PHONE-0450 (I-CTRL).
4615-OP-EXIT.
EXIT.
5010-DO-STRUCTURE.

***************************************************************
* THIS ROUTINE STORES A NEW STRUCTURE RECORD. THE OWNERS       *
* IN THE MANAGES AND REPORTS-TO SETS MUST BE PRESENT          *
* BY THE END OF THE RUN.                                     *
***************************************************************
*****BUILD OWNER OCCURRENCE FOR THE MANAGES SET***************
MOVE 'MANAGES' TO OWNER-ONE-SET.
MOVE -1 TO OWNER-ONE-Serial.
MOVE OG-EMP-MANAGES TO OWNER-ONE-KEY.

*****BUILD OWNER OCCURRENCE FOR SKILL-EXPERTISE SET************
MOVE 'REPORTS-TO' TO OWNER-TWO-SET.
MOVE -1 TO OWNER-TWO-Serial.
MOVE OG-EMP-RPTS-TO TO OWNER-TWO-KEY.

*****BUILD OCCURRENCE OF STRUCTURE RECORD**********************
MOVE OG-STRUCT-CODE TO STRUCTURE-CODE-0460.
MOVE OG-RELATION-DATE TO STRUCTURE-DATE-0460.

5020-STORE-STRUCTURE.
MOVE 'STRUCTURE' TO RECORD-SR-NAME.
MOVE 460 TO RECORD-ID.
MOVE STRUCTURE TO RECORD-DATA.
CALL 'IDMSDBLU' USING OCCURRENCE-DESCRIPTOR
OWNER-DESCRIPTOR-ONE
OWNER-DESCRIPTOR-TWO.
PERFORM DBLU-STATUS.

5090-DS-EXIT.
EXIT.
5510-DO-INSURANCE.

******************************************************************************
* THIS ROUTINE STORES A NEW INSURANCE-PLAN RECORD   *
******************************************************************************

*****BUILD RECORD OCCURRENCE OF INSURANCE-PLAN RECORD***********
MOVE I1-INSPLAN-CODE TO INS-PAN-CODE-0435.
MOVE I1-INSPLAN-CO-NAME TO INS-CO-NAME-0435.
MOVE I2-CO-ADDRESS TO INS-CO-ADDRESS-0435.
MOVE I2-CO-PHONE TO INS-CO-PHONE-0435.
MOVE I3-GROUP-NUM TO GROUP-NUMBER-0435.
MOVE I3-DEDUCT TO DEDUCT-0435.
MOVE I3-MAX-LIFE-COST TO MAXIMUM-LIFE-COST-0435.
MOVE I3-FAM-COST TO FAMILY-COST-0435.
MOVE I3-DEP-COST TO DEP-COST-0435.

5520-STORE-INSURANCE.
MOVE 'INSURANCE-PLAN' TO RECORD-SR-NAME.
MOVE 435 TO RECORD-ID.
MOVE INSURANCE-PLAN TO RECORD-DATA.
CALL 'IDMSDBLU' USING OCCURRENCE-DESCRIPTOR.
PERFORM DBLU-STATUS.

5590-DI-EXIT.
EXIT.
6010-DO-COVERAGE.

******************************************************************************
* THIS MODULE STORES A NEW COVERAGE RECORD. THE OWNER IN *
* THE EMP-COVERAGE SET MUST BE PRESENT BY THE END OF THE *
* RUN. SINCE THIS IS NOT A CALC RECORD THE SERIAL NUMBER *
* RETURNED FROM IDMSDBLU MUST BE SAVED, SO MEMBERS OWNED *
* BY THIS OCCURRENCE CAN REFER TO IT.                              *
******************************************************************************

*****BUILD OWNER OCCURRENCE FOR EMP-COVERAGE SET***************
MOVE 'EMP-COVERAGE' TO OWNER-ONE-SET.
MOVE -1 TO OWNER-ONE-Serial.
MOVE C-EMP-ID TO OWNER-ONE-KEY.

*****BUILD OCCURRENCE OF COVERAGE RECORD***********************
MOVE C-DATAFIELDS TO COVERAGE.
6020-STORE-COVERAGE.
  MOVE ‘COVERAGE’ TO RECORD-SR-NAME.
  MOVE 400 TO RECORD-ID.
  MOVE COVERAGE TO RECORD-DATA.
  CALL ‘IDMSDBLU’ USING OCCURRENCE-DESCRIPTOR
          OWNER-DESCRIPTOR-ONE.
  PERFORM DBLU-STATUS.
  MOVE RECORD-SERIAL TO SAVE-COVERAGE-SERIAL.

6090-DC-EXIT.
EXIT.
6510-DO-DENTAL.

***************************************************************
* THIS ROUTINE STORES A NEW DENTAL-CLAIM RECORD.            *
* THE SERIAL NUMBER OF THE OWNER IN THE COVERAGE-CLAIMS      *
* SET MUST BE OBTAINED AND SAVED PRIOR TO AN ATTEMPT TO      *
* STORE THIS RECORD.                                        *
***************************************************************

*****BUILD OWNER OCCURRENCE FOR COVERAGE-CLAIMS SET************
  MOVE ‘COVERAGE-CLAIMS’ TO OWNER-ONE-SET.
  MOVE -1 TO OWNER-ONE-SERIAL
  MOVE SAVE-COVERAGE-SERIAL TO OWNER-ONE-KEY-SERIAL.

*****BUILD OCCURRENCE OF DENTAL-CLAIM RECORD********************
  MOVE L1-DC-CLAIM-DATE TO CLAIM-DATE-0405.
  MOVE L1-DC-PATIENT-NAME TO PATIENT-NAME-0405.
  MOVE L1-DC-PATIENT-DOB TO PATIENT-BIRTH-DATE-0405.
  MOVE L1-DC-SEX TO PATIENT-SEX-0405.
  MOVE L1-DC-REL-TO-EMP TO RELATION-TO-EMPLOYEE-0405.
  MOVE L1-DC-DENTIST-NAME TO DENTIST-NAME-0405.
  MOVE L2-DC-DENTIST-ADDRESS TO DENTIST-ADDRESS-0405.
  MOVE L2-DC-DENTIST-LIC-NUM TO DENTIST-LICENSE-NUMBER-0405.
  MOVE I-CTRL TO NUMBER-OF-PROCEDURES-0405.

6520-STORE-DENTAL.
  MOVE ‘DENTAL-CLAIM’ TO RECORD-SR-NAME.
  MOVE 405 TO RECORD-ID.
  MOVE DENTAL-CLAIM TO RECORD-DATA.
  CALL ‘IDMSDBLU’ USING OCCURRENCE-DESCRIBTOR
          OWNER-DESCRIPTOR-ONE.
  PERFORM DBLU-STATUS.

6590-DDN-EXIT.
EXIT.
7010-DO-HOSPITAL.

******************************************************************************
* THIS ROUTINE STORES A NEW HOSPITAL-CLAIM RECORD.            *
* THE SERIAL NUMBER OF THE OWNER IN THE COVERAGE-CLAIMS      *
* SET MUST BE OBTAINED AND SAVED PRIOR TO AN ATTEMPT TO      *
* STORE THIS RECORD.                                        *
******************************************************************************

*****BUILD OWNER OCCURRENCE FOR COVERAGE-CLAIMS SET************
  MOVE ‘COVERAGE-CLAIMS’ TO OWNER-ONE-SET.
  MOVE -1 TO OWNER-ONE-SERIAL
  MOVE SAVE-COVERAGE-SERIAL TO OWNER-ONE-KEY-SERIAL.

*****BUILD OCCURRENCE OF HOSPITAL-CLAIM RECORD********************
  MOVE H1-HC-CLAIM-DATE TO CLAIM-DATE-0430.
  MOVE H1-HC-PATIENT-NAME TO PATIENT-NAME-0430.
  MOVE H1-HC-PATIENT-DOB TO PATIENT-BIRTH-DATE-0430.
  MOVE H1-HC-SEX TO PATIENT-SEX-0430.
  MOVE H1-HC-REL-TO-EMP TO RELATION-TO-EMPLOYEE-0430.
  MOVE H1-HC-HOSP-NAME TO HOSPITAL-NAME-0430.
  MOVE H2-HC-HOSP-ADDRESS TO HOSP-ADDRESS-0430.
  MOVE H2-HC-ADMIT-DATE TO ADMIT-DATE-0430.
MOVE H2-HC-DISCH-DATE TO DISCHARGE-DATE-0430.
MOVE H3-HC-DIAGNOSIS TO DIAGNOSIS-0430 (1).
MOVE H4-HC-DIAGNOSIS TO DIAGNOSIS-0430 (2).
MOVE H5-HC-WARD-DAYS TO WARD-DAYS-0430.
MOVE H5-HC-WARD-RATE TO WARD-RATE-0430.
MOVE H5-HC-WARD-TOTAL TO WARD-TOTAL-0430.
MOVE H5-HC-SEMI-DAYS TO SEMI-DAYS-0430.
MOVE H5-HC-SEMI-RATE TO SEMI-RATE-0430.
MOVE H5-HC-SEMI-TOTAL TO SEMI-TOTAL-0430.
MOVE H5-HC-DEL-COST TO DELIVERY-COST-0430.
MOVE H5-HC-ANESTH-COST TO ANESTHESIA-COST-0430.
MOVE H5-HC-LAB-COST TO LAB-COST-0430.
7020-STORE-HOSPITAL.
MOVE ‘HOSPITAL-CLAIM’ TO RECORD-SR-NAME.
MOVE 430 TO RECORD-ID.
MOVE HOSPITAL-CLAIM TO RECORD-DATA.
CALL ‘IDMSDBLU’ USING OCCURRENCE-DESCRIPTOR
OWNER-DESCRIPTOR-ONE.
PERFORM DBLU-STATUS.
7090-DH-EXIT.
EXIT.
7510-DO-NONHOSP.

**************************************************************
* THIS ROUTINE STORES A NEW NON-HOSP-CLAIM RECORD.         *
* THE SERIAL NUMBER OF THE OWNER IN THE COVERAGE-CLAIMS      *
* SET MUST BE OBTAINED AND SAVED PRIOR TO AN ATTEMPT TO      *
* STORE THIS RECORD.                                        *
**************************************************************

*****BUILD OWNER OCCURRENCE FOR COVERAGE-CLAIMS SET**********

MOVE ‘COVERAGE-CLAIMS’ TO OWNER-ONE-SET.
MOVE -1 TO OWNER-ONE-SERIAL
MOVE SAVE-COVERAGE-SERIAL TO OWNER-ONE-KEY-SERIAL.

*****BUILD OCCURRENCE OF NON-HOSP-CLAIM RECORD**************

MOVE N1-NC-CLAIM-DATE TO CLAIM-DATE-0445.
MOVE N1-NC-PATIENT-NAME TO PATIENT-NAME-0445.
MOVE N1-NC-PATIENT-DOB TO PATIENT-BIRTH-DATE-0445.
MOVE N1-NC-SEX TO PATIENT-SEX-0445.
MOVE N1-NC-REL-TO-EMP TO RELATION-TO-EMPLOYEE-0445.
MOVE N1-NC-PHYS-NAME TO PHYSICIAN-NAME-0445.
MOVE N2-NC-PHYS-ADDRESS TO PHYSICIAN-ADDRESS-0445.
MOVE N2-NC-PHYS-ID TO PHYSICIAN-ID-0445.
MOVE N3-NC-DIAGNOSIS TO DIAGNOSIS-0445 (1).
MOVE N4-NC-DIAGNOSIS TO DIAGNOSIS-0445 (2).
MOVE I-CTRL TO NUMBER-OF-PROCEDURES-0445.

7520-STORE-NONHOSP.
MOVE ‘NON-HOSP-CLAIM’ TO RECORD-SR-NAME.
MOVE 445 TO RECORD-ID.
MOVE NON-HOSP-CLAIM TO RECORD-DATA.
CALL ‘IDMSDBLU’ USING OCCURRENCE-DESCRIPTOR
OWNER-DESCRIPTOR-ONE.
PERFORM DBLU-STATUS.
7590-DN-EXIT.
EXIT.

**************************************************************
* CLOSE OUT LOAD PROGRAM OPERATIONS HERE.                      *
* DISPLAY APPROPRIATE RUN-TIME STATISTICS FROM PROGRAM         *
* AND DATABASE SYSTEM; THEN CALL IDMSDBLU WITH A -1           *
* IN RECORD-ID TO CLOSE HIS FILES AND PUT OUT A CONTROL       *
* RECORD.                                                    *
**************************************************************

9999-END.
DISPLAY SUM-CARDS-IN 'CARDS'.
DISPLAY SUM-TRANSACTIONS 'TRANSACTIONS'.
MOVE -1 TO RECORD-ID.
CALL 'IDMSDBLU' USING OCCURRENCE-DESCRIPTOR.

GOBACK.

**********************************************************************
<table>
<thead>
<tr>
<th>DBLU-STATUS</th>
<th>SECTION.</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF RECORD-LOAD-STATUS NOT = '0000'</td>
<td></td>
</tr>
<tr>
<td>DISPLAY 'LOAD STATUS ------ ' RECORD-LOAD-STATUS</td>
<td></td>
</tr>
<tr>
<td>DISPLAY 'RECORD NAME ------ ' RECORD-SR-NAME</td>
<td></td>
</tr>
<tr>
<td>DISPLAY 'RECORD ID ------ ' RECORD-ID</td>
<td></td>
</tr>
<tr>
<td>DISPLAY 'RECORD SERIAL NO.-- ' RECORD-SERIAL</td>
<td></td>
</tr>
<tr>
<td>DISPLAY 'SUGGESTED PAGE ---- ' RECORD-SUGGESTED-PAGE</td>
<td></td>
</tr>
<tr>
<td>DISPLAY 'RECORD DATA ------ ' RECORD-DATA-REDEF</td>
<td></td>
</tr>
<tr>
<td>DISPLAY '******************* '.</td>
<td></td>
</tr>
</tbody>
</table>

DBLU-STATUS-EXIT.
EXIT.
IDMSRSTT Macro Statements

IDMSRSTT macro statements define the changes to be made to the database during a restructure operation. The statements reflect the information in two schemas:

- **An old schema** that describes the database before restructuring
- **A new schema** that describes the database after restructuring

You can code the IDMSRSTT macro statements sectionally, or you can use the IDMSRSTC utility to generate the statements automatically. In either case, you must assemble the statements into a base restructuring table, which is then used by the RESTRUCTURE and RESTRUCTURE CONNECT utility statements.

- For more information about generating the IDMSRSTT macro statements automatically, see IDMSRSTC (see page 349).
- For more information on restructuring a database, see RESTRUCTURE (see page 241) and RESTRUCTURE CONNECT (see page 248).

**Statement descriptions**

You assemble a base restructuring table from the following statements:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDMSRSTT BUFSIZE</td>
<td>One for a base restructuring table</td>
</tr>
<tr>
<td>IDMSRSTT RECNAME</td>
<td>One for each record type being modified</td>
</tr>
<tr>
<td>IDMSRSTT SETPTR</td>
<td>For each record type being modified: \nOne for each pointer in the record if any pointers are being changed \nOne that specifies ALL if no pointers are being changed</td>
</tr>
<tr>
<td>IDMSRSTT FIELD</td>
<td>For each record type being modified: \nOne for each field or group of contiguous fields if the record is variable length or if the record is fixed length and one or more fields are being changed \nOne that specifies ALL if the record is fixed length and no fields are being changed</td>
</tr>
<tr>
<td>IDMSRSTT END</td>
<td>One for each base restructuring table</td>
</tr>
<tr>
<td>END</td>
<td>One for each base restructuring table</td>
</tr>
</tbody>
</table>

**Coding considerations**

Use the following conventions when reviewing, modifying, or sectionally coding IDMSRSTT macro statements:

- Code each statement in uppercase on a separate line, beginning in column 2 or greater
IDMSRSTT BUFSIZE

The IDMSRSTT BUFSIZE statement specifies the sizes of the record buffers to be used during RESTRUCTURE utility processing.

Syntax

```plaintext
IDMSRSTT BUFSIZE = (old-buffer-size, new-buffer-size )
```

Parameter

- **old-buffer-size**
  Specifies the size, in bytes, of a buffer large enough to hold the largest restructured record, as defined in the old schema.

- **new-buffer-size**
  Specifies the size, in bytes, of a buffer large enough to hold the largest restructured record, as defined in the new schema.
Usage

IDMSRSTT BUFSIZE must be the first macro statement coded for each base restructuring table you assemble.

**IDMSRSTC buffer-size values**

The IDMSRSTC utility calculates the values of *old-buffer-size* and *new-buffer-size* to be the length of the largest record in the applicable schema, rounded up to the nearest multiple of four. The record length equals:

- The length of the prefix
- Plus the length of the data portion
- Plus, for variable-length records, an 8-byte overhead

**Estimating buffer-size values**

When coding the IDMSRSTT BUFSIZE statement sectionly, make a generous estimate for the *old-buffer-size* and *new-buffer-size* parameters. For example, round the calculated record length up to the nearest multiple of 100.

**IDMSRSTT RECNAME**

The IDMSRSTT RECNAME statement identifies a record type that is being restructured and provides new format and length information for the record, if applicable. The IDMSRSTT RECNAME statement also names database procedures to be executed during the restructure process.

This article describes the following information:

- Syntax (see page 480)
- Parameters (see page 481)
- Usage (see page 482)

**Syntax**

```
IDMSRSTT RECNAME = record-name

;MINLEN = { (min-root-length,min-fragment-length,max-data-length) FIXED

DCT = dctname

;NUPROCS = ( procedure-name )
```
Parameters

- **RECCNAME = record-name**
  Specifies the name of the record being restructured.

- **MINLEN**
  New format and/or length information for the record.

- **min-root-length**
  Applies only to variable-length records and fixed-length compressed records.
  The minimum root length, in bytes, of the record, rounded up to the nearest multiple of four,
  with the following qualifications:
  - This value does not include the four-byte variable-length indicator for variable-length records
    nor the record prefix.
  - If the record is being changed from fixed-length uncompressed, min-root-length must be at
    least four.
  - If the minimum root length is not being changed, min-root-length must be zero.

- **min-fragment-length:**
  The minimum fragment length, in bytes, for the record, not including the fragment prefix. Must
  be at least four.

- **max-data-length:**
  The maximum data length, in bytes, for the record, not including the four-byte variable-length
  indicator, rounded up to the nearest multiple of four.

- **FIXED**
  Specifies that the record is being changed from variable-length or fixed-length compressed to
  fixed-length uncompressed.

- **DCT =**
  Specifies a PressPack DCT to use when:
  - Converting the record from uncompressed to PressPack compressed
  - Converting a PressPack compressed record from another PressPack DCT to the specified
    PressPack DCT.

- **dctname:**
  The name of the PressPack DCT.

- **NUPROC =**
  Specifies one or more database procedures to be executed during the restructure process. The
  procedures are executed just before the RESTRUCTURE utility uses a MODIFY statement to
  rewrite the restructured record to the database.

- **procedure-name:**
  Specifies the name of the procedure.
Usage

Account for all changing records

Code one RECNAME statement for each:

- Record being restructured

- Procedure to be executed before RESTRUCTURE utility uses a MODIFY statement to rewrite the restructured record

Length parameters

All lengths are specified in bytes, rounded up to the nearest multiple of four.

Compressing records with DCT =

Records to be converted to PressPack compressed must be variable length records, or have a new format specified by the MINLEN parameter. You cannot specify MINLEN = FIXED for such records.

Records being converted to PressPack compressed should not have any Before Modify procedures defined that expect to see uncompressed data. The record will already be compressed before the Before procedures are called.

RESTRICTURE will issue a warning if it encounters Before Modify procedures, but it will continue processing.

Uncompressing PressPack compressed records

To convert records from PressPack compressed records to fixed uncompressed, specify MINLEN = FIXED.

When to use NUPROCS =

List all procedures to be executed for a record during the restructure process before a MODIFY statement, as follows:

- A procedure specified in the new schema should be included if it is not included in the old schema, as long as the procedure is to be executed before a MODIFY statement. For example, if CALL IDMSSCOMP BEFORE MODIFY is specified for the record in the new schema, include IDMSSCOMP.

**Note:** If a procedure is specified in the new schema but is not called before a MODIFY statement, do not include the procedure here. For example, if CALL IDMSDCOM AFTER GET is specified for the record in the new schema, IDMSDCOM should not be included here.
A procedure that is not specified in the new schema should be included if the procedure is to be executed before a MODIFY statement during the restructure process; such procedures must be added sectionly to the IDMSRSTT macro statements generated by IDMSRSTC. For example, a procedure that initializes a new data item specified in an IDMSRSTT FIELD statement for the record should be included.

**IDMSRSTT SETPTR**

The IDMSRSTT SETPTR statement specifies pointer positions for a restructured record. Using the IDMSRSTT SETPTR statement, you can:

- Add new pointers
- Delete existing pointers
- Initialize existing pointers
- Copy existing pointers to new positions

This article describes the following information:

- Syntax (see page 483)
- Parameters (see page 483)
- Usage (see page 484)

**Syntax**

```
IDMSRSTT SETPTR =
```

```
( old-position, new-position, set-name )
```

```
ALL
```

**Parameters**

- **old-position**: Specifies the position of the pointer in the original record.
- ***:** Specifies that the pointer is new and it's being added to the owner record.

---

**Note:** If a new pointer is being added to the member record, do not specify `old-position` or `*`; omit the parameter entirely.
new-position:
Specifies the new position for a new pointer or for a pointer being moved. If the pointer is being deleted, do not specify a new position.

set-name:
The name of an existing set for which prior pointers are being added.

Note: Do not use this parameter if owner pointers are being added to an existing set.

ALL
Specifies that no pointers in the record are being added, deleted, or changed.

Usage

Accounting for all pointers
The IDMSRSTT SETPTR statements you code for a record must account for all the pointers in the record:

- If you are adding, modifying, or deleting any pointers in the record, code one IDMSRSTT SETPTR statement for each pointer in the record.
- If you are not adding, modifying, or deleting any pointers in the record, code a single IDMSRSTT SETPTR=ALL statement.

Calculating pointer positions
The first user pointer position in a record is 1.

Do not count system-maintained pointers (for example, those for the CALC set) when calculating pointer positions.

Order of changes
You can modify pointers in any order and with any sequence of modifications. For example, you can add a new pointer in position 4 before or after deleting an existing pointer in position 2.

Overwriting existing pointers
When you add a new pointer or copy an existing pointer to a position that already contains a pointer, the pointer in the position is overwritten.

IDMSRSTT FIELD
The IDMSRSTT FIELD statement identifies the data fields in a restructured record. Using the IDMSRSTT FIELD statement, you can:
Add new data fields

Delete existing data fields

Move existing data fields to new positions

This article describes the following information:

- **Syntax** (see page 485)
- **Parameters** (see page 485)
- **Usage** (see page 486)

### Syntax

```
IDMSRSTT FIELD =

( field-specification )

.OCC = occurrence-count

Expansion of field-specification

.new-field-value

.old-displacement

/new-displacement

.field-length

/NEW

/CTRL
```

### Parameters

- **field-specification**
  Specifies a field or group of contiguous fields.
  For the purposes of IDMSRSTT, a group of contiguous fields, which stays together as a group and undergoes changes as a group, can be treated as a single field.
  See the following expanded parameters.

- **.OCC =**
  Identifies the field as occurring more than once in the restructured record.

- **occurrence-count:**
  The number of occurrences of the field, as defined by an OCCURS clause for the field in the new schema.
  For a contiguous group of fields, they must all be defined by identical OCCURS clauses in the new schema.
• DEP
  Indicates that the field is defined by an OCCURS DEPENDING ON clause in the new schema.
  For a contiguous group of fields, they must all be defined by identical OCCURS DEPENDING ON
  clauses in the new schema.

• ALL
  Specifies that all data items in the original record are duplicated in the new record, with no
  additions, deletions, or changes and the control length of the record is not being changed. The
  control length of a record can change if a set is being changed from sorted to unsorted, unsorted
  to sorted, or if a sorted set, index key, or CALC key is being added or removed from the record.

• old-displacement:
  For previously existing fields only.
  Specifies the beginning location of the field in the original record.

• new-field-value:
  For new fields only.
  Specifies the initial value of the new field, expressed as an Assembler constant.
  See "Usage" in this section.

• new-displacement:
  Specifies the beginning location of the field in the restructured record.

• field-length:
  Specifies the length, in bytes, of the field.
  For a field that occurs more than once, field-length specifies the length of a single occurrence.

• NEW
  Specifies that the field is a new field being added to the restructured record.

• CTRL
  Specifies that the field is a control field (CALC key, sort key, or index key). This parameter can be
  used for any type of record, but must be specified if ALL is not specified, control fields exist within
  the record, and the record is:

  - Fixed-length compressed
  - Variable-length (compressed or uncompressed)
  - Associated with a database procedure

  If specified, CTRL need only be included on the last control field within the record. If you choose,
  you can specify it on all control fields.

Usage

Contiguous fields
If a contiguous group of fields remains contiguous and internally unchanged in all occurrences, it can be treated as if it were a single field. Therefore, field-length, old-displacement, and new-displacement can refer to the entire group.

If any part of the group is changed, or occurring multiply without maintaining the same relationship with the rest of the group, then it must be treated separately.

**Accounting for all data fields**

The IDMSRSTT FIELD statements you code for a record must account for all the data fields in the record:

- If you are adding, modifying, or deleting any data fields in the record, or if the control field of the record is changing, code one IDMSRSTT FIELD statement for each field or group of contiguous fields in the record.

- If you are not adding, modifying, or deleting any data fields in the record, and the control length of the record is not changing, code a single IDMSRSTT FIELD=ALL statement.

**Calculating data field positions**

The first byte of data in a record is at location 1. When calculating field positions for variable-length records, do not count the 4-byte variable-length indicator maintained at the beginning of the data portion.

**Specifying new field values**

The value for new-field-value is expressed as an Assembler constant (for example, CL2 ' ', indicating a 2-byte field initialized to spaces).

For new fields that have not been assigned an initial value in the new schema, IDMSRSTC supplies an initial value based on the usage specification in the schema, as follows:

<table>
<thead>
<tr>
<th>Usage</th>
<th>Initial value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISPLAY, (PIC X), BIT, or POINTER</td>
<td>CLX ' '</td>
</tr>
<tr>
<td>DISPLAY, (PIC 9)</td>
<td>xCL1'0'</td>
</tr>
<tr>
<td>COMP (BINARY)</td>
<td>XLx'0'</td>
</tr>
<tr>
<td>COMP-1 (SHORT-POINT)</td>
<td>F'0'</td>
</tr>
<tr>
<td>COMP-2 (LONG-POINT)</td>
<td>D'0'</td>
</tr>
<tr>
<td>COMP-3 (PACKED)</td>
<td>PLx'0'</td>
</tr>
</tbody>
</table>

X is the length, in bytes, of the field.

**Expansion to the right of a decimal point**

The restructure utility as a general rule cannot expand a field to the right of a decimal point. To do this, the standard procedure would be to first add the needed bytes to the left of the field. After the RESTRUCTURE has been performed, a user-written program must modify each affected record by multiplying the expanded fields by the proper factor to realign the decimal point to the new position.
IDMSRSTT END

The IDMSRSTT END statement terminates the restructuring information.

IDMSRSTT END must be the next to last macro statement coded for each base restructuring table you assemble.

Syntax

```
IDMSRSTT END
```

END (IDMSRSTT)

The END statement terminates the assembler input.

END must be the last macro statement coded for each base restructuring table you assemble.

Syntax

```
END
```
Common Facilities for Distributed Transactions

This section discusses the recovery utilities that report on distributed transactions and that support the use of a section recovery control file for use with the two-phase commit feature. This section describes these common enhancements as they apply to the following recovery utility statements:

- EXTRACT JOURNAL
- FIX ARCHIVE
- MERGE ARCHIVE
- PRINT JOURNAL
- ROLLBACK
- ROLLFORWARD

For more information, see the following topics:
- Reporting on Distributed Transactions (see page 489)
- Manual Recovery Input Control File (see page 491)
- Manual Recovery Output Control File (see page 492)
- Execution JCL Changes (see page 493)

Reporting on Distributed Transactions

A distributed transaction journal record consists of a fixed portion and up to three variable arrays of data. The fixed portion contains the distributed transaction identifier (DTRID) and a local branch ID (BID), which identifies an individual branch of the distributed transaction. The fixed portion can be followed by any of the following:

- A list of local transaction identifiers (LIDs), one for each branch of the transaction that made local database changes
- A list of external transaction identifiers if the transaction is known externally by another identifier, such as an XA XID or an RRS URID
- A list of interests that other resource or transaction managers have in the distributed transaction

The recovery utilities report some or all of the above information in their detailed report and list distributed transactions that were incomplete at stop time in their summary report.
The following example shows the output that is produced by PRINT JOURNAL REPORT FULL when it encounters a typical DCOM record. If the REPORT TERSE option is specified, neither external transaction identifiers nor resource manager interests are included. Other recovery utilities show similar information.

```
NODE SYSTEM74 DTRID-BID SYSTEM74::01650C9509CE38A3-01650C90A4CDA0BD DCOM
LOC_ID 10016 PGM_ID PROCDISM
RMS URID BBDEBCA5784A1650C90A4CDA0BD
RRS URID B8DEBCA5784A1650C90A4CDA0BD
RRS URID B8DEBCA5784A1650C90A4CDA0BD
RRS URID B8DEBCA5784A1650C90A4CDA0BD
RRS URID B8DEBCA5784A1650C90A4CDA0BD
RRS URID B8DEBCA5784A1650C90A4CDA0BD
RRS URID B8DEBCA5784A1650C90A4CDA0BD
RRS URID B8DEBCA5784A1650C90A4CDA0BD
RRS URID B8DEBCA5784A1650C90A4CDA0BD
RRS URID B8DEBCA5784A1650C90A4CDA0BD
RRS URID B8DEBCA5784A1650C90A4CDA0BD
RRS URID B8DEBCA5784A1650C90A4CDA0BD
RRS URID B8DEBCA5784A1650C90A4CDA0BD
```

A brief description of the report's contents follows. For an in-depth discussion of the meaning of this report, see CA IDMS Database Administration Section.

- **Node SYSTEM74** identifies the name of the system that produced the journal entry, in this case, **SYSTEM74**.

- **DTRID-BID** SYSTEM74::01650C9509CE38A3-01650C90A4CDA0BD identifies the DTRID and the BID of the top-level branch of the distributed transaction for which the DCOM record was written. The DTRID is **SYSTEM74::01650C9509CE38A3** and the BID is **01650C90A4CDA0BD**.

- **DCOM** is the type of distributed transaction journal record that is being reported.

- **LOC_ID** 10016 identifies the work done by a local transaction branch that is included in the distributed transaction. In this case, the local identifier is **10016**.

- **PGM_ID** PROCDISM identifies the name of the application program that started the local transaction branch. In this case, the program is **PROCDISM**.

- **RRS URID** BBDEBCA5784A1650C90A4CDA0BD identifies the transaction, as it is known externally.

- **RM NAME** SYSTEM74::RRS_RMI identifies a resource manager that has registered an interest in the distributed transaction. In this case, the resource manager is **RRS**.

- **TYPE RRS** indicates that the RM type is **RRS**.

- **ROLE SDSRM** indicates that this interest is the controlling interest for the transaction, and therefore RRS is the transaction's coordinator.

- **STATE InDoubt** indicates the interest's state. In this case, the interest is in an **InDoubt** state.

- **FLG1/2** 0001 displays flags that are used to restart the transaction following a system failure.

- **EXITS** 40 0034000000000000 shows the exits that have been registered by the resource manager and the responses returned by the exits that have already been called during the life of the transaction.
- `D9D9E240C2...` shows the data (in hex and character format) that the resource manager wishes preserved should it be necessary to restart the transaction following a system failure. This information will vary depending on the resource manager that registered the interest.

- `RM NAME SYSTEM73::DSI_CLI` identifies a resource manager that has registered an interest in the distributed transaction. In this case, the resource manager is a CA IDMS system named `SYSTEM73`.

- `TYPE IDMS` indicates the type of the resource manager.

- `ROLE CRM` indicates that this interest is not a controlling interest for the transaction. Therefore, the associated resource manager (`SYSTEM73`) is a participant in the transaction. 

- `FLG1/2 0000` displays flags that are used to restart the transaction following a system failure.

- `EXITS 76 000000000000000000000000` shows the exits that have been registered by the resource manager and the responses returned by the exits that have already been called during the life of the transaction.

- `E2E8E2E3C5...` shows the date (in hex and character format) that the resource manager wishes to have preserved if it is necessary to restart the transaction following a system failure. This information will vary depending on the resource manager that registered the interest.

### Manual Recovery Input Control File

A section recovery input control file can be used to explicitly specify whether an InDoubt distributed transaction should be committed or backed out. Its use is optional, but if included in a utility's execution JCL, it will be used as input to the following recovery operations:

- `EXTRACT JOURNAL` (unless `ALL` is specified)
- `FIX ARCHIVE`
- `MERGE ARCHIVE` (if `COMPLETE` is specified)
- `PRINT JOURNAL`
- `ROLLBACK`
- `ROLLFORWARD` (unless `ALL` is specified)

The Input Control file only affects InDoubt transactions. It will have no effect on transactions that were in a Commit or BackOut state; or forgotten.

If the Input Control File is not used to commit or back out an InDoubt transaction, its state will remain in doubt. Its updates will still be applied or remain applied on the database and no journal records will be generated for that transaction. The updates will be complete and the database intact, that is, no broken pointers. Provided the database is not used and updated, the InDoubt transactions can still be backed out or formally committed in a later recovery job.
The file contains 80-byte records whose format is:

<DTrid><Action>

Where <Dtrid> is a 26-character display-format DTRID and <Action> is either COMMIT or BACKOUT. If more than one record specifies the same DTRID value, all but the last one are ignored.

The following example specifies that the transaction identified by DTRID SYSTEM74::01650C9509CE38A3 should be backed out:

SYSTEM74::01650C9509CE38A3 BACKOUT

If section control input entries are used in a recovery operation that creates an output journal file (FIX ARCHIVE, EXTRACT JOURNAL, and MERGE ARCHIVE), then additional distributed transaction journal records will be written to the output file to complete the transaction in the specified way.

The following is a sample of the report generated by FIX ARCHIVE. It lists entries in the section recovery input control file and shows the effect of those entries in its summary report. In this example, the distributed transaction identified by CICSCICS::B8AD18E5A9BF0F41 is committed by the generation of new DCOM and DFGT journal records.

Input Control Records:
CICSCICS::B8AD18E5A9BE1300 BACKOUT
CICSCICS::B8AD18E5A9BF0F41 COMMIT

Incomplete Distributed Transactions At Stop Time:

<table>
<thead>
<tr>
<th>NODE</th>
<th>DTRID-BID</th>
<th>STATE</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>***</td>
<td>SYSTEM74 CICSCICS::B8AD18E5A9BF0F41-016507A67C2E6D53 InDoubt Commit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>*GEN</td>
<td>SYSTEM74 DTRID-BID CICSCICS::B8AD18E5A9BF0F41-016507A67C2E6D53 DCOM</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LOC ID 28 PGM IE CICSDML1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>*GEN</td>
<td>SYSTEM74 DTRID-BID CICSCICS::B8AD18E5A9BF0F41-016507A67C2E6D53 DFGT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Manual Recovery Output Control File

Since a section recovery control file is an 80-byte card image file, you can create it with a text editor. A prototype control file can also be optionally created by any of the following recovery operations if a section recovery output control file is included in its execution JCL:

- EXTRACT JOURNAL
- FIX ARCHIVE
- MERGE ARCHIVE
- PRINT JOURNAL
- ROLLFORWARD (unless FROM EXTRACT is specified)

When a control file is generated, an entry is created for every distributed transaction whose final state is InDoubt. Automatically generated entries always specify that the transaction should be backed out. It is expected that the resulting file will be edited prior to using it as input to a recovery operation.
Execution JCL Changes

Manual recovery control files are optional, so no execution JCL changes are necessary unless their use is desired.

To use a section recovery input control file, include a CTRLIN file definition or DD statement in the IDMSBCF execution JCL. To use a section recovery output control file, include a CTRLOUT file definition or DD statement in the IDMSBCF execution JCL. The format of both of these files is fixed block with a record length of 80.