Notes

Before using this information and the product it supports, be sure to read the general information under “Notices” at the end of this information.

Subsequent editions of this PDF will not be delivered in IBM Publications Center. Always download the latest edition from Db2 11 for z/OS Product Documentation.

April 1, 2019 edition

This edition applies to Db2 11 for z/OS (product number 5615-DB2), Db2 11 for z/OS Value Unit Edition (product number 5697-P43), Db2 Utilities Suite for z/OS, Version 11 (product number 5655-W87), and to any subsequent releases until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

Specific changes are indicated by a vertical bar to the left of a change. A vertical bar to the left of a figure caption indicates that the figure has changed. Editorial changes that have no technical significance are not noted.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
Contents

About this information ........................................... xv
Who should read this information .................................. xv
Db2 Utilities Suite for z/OS ........................................... xv
Terminology and citations ............................................. xvi
Accessibility features for Db2 11 for z/OS ......................... xvi
How to send your comments ......................................... xvii
Naming conventions used in this information ..................... xvii
How to read syntax diagrams ....................................... xx

Part 1. Basic information about the Db2 utilities .................. 1
Chapter 1. Db2 utilities packaging ..................................... 3
Chapter 2. Privileges and authorization IDs for Db2 utilities .......... 5
Chapter 3. Utilities that can be run on declared temporary objects .... 7
Chapter 4. Effect of utilities on objects that have the DEFINE NO attribute .... 9
Chapter 5. Effect of utilities on encrypted data ....................... 11

Part 2. Db2 online utilities ............................................. 13
Chapter 6. Invoking Db2 online utilities ....................... 15
Data sets that online utilities use .................................. 16
Utility control statements ........................................... 18
Invoking an online utility by using the DB2 Utilities panel in DB2I ................................. 21
DB2 Utilities panel .................................................. 23
Data Set Names panel ................................................ 24
Control Statement Data Set Names panel........................ 25
Invoking a Db2 utility by using the DSNU CLIST command in TSO .......... 26
  DSNU CLIST command syntax ................................ 27
  DSNU CLIST command output ................................ 32
The supplied JCL procedure (DSNUPROC) for invoking a Db2 online utility .......... 33
Invoking a Db2 online utility by creating the JCL data set yourself ........... 36

Chapter 7. Monitoring and controlling online utilities .......... 39
Monitoring utilities .................................................. 39
Monitoring processor use by utilities ................................ 40
Subsystem parameters for refining DFSMSdss COPY operation with utilities ........ 40
Determining why a utility failed to complete ....................... 42
Terminating an online utility ....................................... 43
Restarting an online utility ......................................... 44
  Overriding the default utility restart behavior by using the RESTART parameter .... 47
  Restarting a utility after the output data set is full .................. 48
  How utilities restart with templates ................................ 49
  How utilities restart with lists ................................... 49

Chapter 8. BACKUP SYSTEM ......................................... 51
Syntax and options of the BACKUP SYSTEM control statement .......... 52
Before running BACKUP SYSTEM .................................. 54
  Copy pools ......................................................... 54

© Copyright IBM Corp. 1983, 2017
Data sets that BACKUP SYSTEM uses ...................................................... 55
Concurrency and compatibility for BACKUP SYSTEM .......................... 56
Dumping a fast replication copy to tape .................................................. 56
Backups of log copy pools ................................................................. 57
Termination or restart of BACKUP SYSTEM ........................................ 57
Sample BACKUP SYSTEM control statements ..................................... 57

Chapter 9. CATENFM. .................................................................... 61
Syntax and options of the CATENFM control statement ......................... 61
Before converting the catalog ............................................................. 62
  Data sets that CATENFM uses when converting the catalog .................. 62
Concurrency and compatibility for CATENFM ...................................... 63
Converting to new-function mode ....................................................... 63
Termination or halt of CATENFM ........................................................ 64

Chapter 10. CATMAINT ................................................................. 65
Updating the catalog for installation or migration to Db2 11 ...................... 68
Renaming the owner, creator, and schema of database objects ................. 69
Changing the ownership of objects from an authorization ID to a role ...... 69
Changing the catalog name used by storage groups or index spaces and table spaces .................................................. 70
Identifying invalidated packages after the owner, creator, or schema name of an object is renamed ................................................. 70

Chapter 11. CHECK DATA ............................................................ 73
Syntax and options of the CHECK DATA control statement .................. 74
Before running CHECK DATA ............................................................ 85
  Data sets that CHECK DATA uses ..................................................... 86
Concurrency and compatibility for CHECK DATA ................................ 90
Exception tables for the CHECK DATA utility ..................................... 92
Exception processing for tables with auxiliary columns ....................... 93
Specifying the scope of CHECK DATA ............................................... 93
How violations are identified ............................................................ 94
Detection and correction of constraint violations .................................. 95
CHECK DATA XML error detection ................................................... 95
Correcting XML data after running CHECK DATA ................................ 96
Resetting CHECK-pending status ....................................................... 97
LOB column errors ........................................................................ 97
Resetting auxiliary CHECK-pending status ....................................... 98
Termination and restart of CHECK DATA .......................................... 99
Sample CHECK DATA control statements .......................................... 99

Chapter 12. CHECK INDEX ............................................................. 103
Syntax and options of the CHECK INDEX control statement ................. 104
Data sets that CHECK INDEX uses .................................................... 108
  Shadow data sets for CHECK INDEX ............................................... 109
Concurrency and compatibility for CHECK INDEX ............................... 111
Single logical partitions .................................................................. 113
Indexes in parallel ........................................................................... 114
Reviewing CHECK INDEX output ...................................................... 117
Termination or restart of CHECK INDEX ............................................ 118
Correcting XML data after running CHECK INDEX ............................... 118
Sample CHECK INDEX control statements ......................................... 118

Chapter 13. CHECK LOB ............................................................... 121
Syntax and options of the CHECK LOB control statement ....................... 122
Before running CHECK LOB ............................................................. 125
  Data sets that CHECK LOB uses ....................................................... 125
Concurrency and compatibility for CHECK LOB ................................ 128
How CHECK LOB identifies violations .............................................. 129
Removing CHECK-pending status for a LOB table space ....................... 129
Resolving media failure .................................................................. 129
<table>
<thead>
<tr>
<th>Chapter 14. COPY</th>
<th>133</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax and options of the COPY control statement</td>
<td>135</td>
</tr>
<tr>
<td>Before running COPY</td>
<td>147</td>
</tr>
<tr>
<td>Data sets that COPY uses</td>
<td>147</td>
</tr>
<tr>
<td>Concurrency and compatibility for COPY</td>
<td>150</td>
</tr>
<tr>
<td>Full image copies</td>
<td>153</td>
</tr>
<tr>
<td>Incremental image copies</td>
<td>154</td>
</tr>
<tr>
<td>Multiple image copies</td>
<td>155</td>
</tr>
<tr>
<td>FlashCopy image copies</td>
<td>157</td>
</tr>
<tr>
<td>Copies of lists of objects</td>
<td>161</td>
</tr>
<tr>
<td>Using more than one COPY statement</td>
<td>163</td>
</tr>
<tr>
<td>Copying partitions or data sets simultaneously</td>
<td>163</td>
</tr>
<tr>
<td>Copies of partition-by-growth table spaces</td>
<td>164</td>
</tr>
<tr>
<td>Copies of XML table spaces</td>
<td>164</td>
</tr>
<tr>
<td>Copying catalog and directory objects</td>
<td>165</td>
</tr>
<tr>
<td>Make copies of XML schema repository objects</td>
<td>165</td>
</tr>
<tr>
<td>Copies of Indexes</td>
<td>166</td>
</tr>
<tr>
<td>Using DFSMSdss concurrent copy</td>
<td>166</td>
</tr>
<tr>
<td>Specifying conditional image copies</td>
<td>168</td>
</tr>
<tr>
<td>Allocation of sequential image copy data sets</td>
<td>169</td>
</tr>
<tr>
<td>Preparing for recovery by using the COPY utility</td>
<td>170</td>
</tr>
<tr>
<td>Improving performance of the COPY utility</td>
<td>171</td>
</tr>
<tr>
<td>Generation data group definitions for the COPY utility</td>
<td>171</td>
</tr>
<tr>
<td>Using Db2 with DFSMS products</td>
<td>172</td>
</tr>
<tr>
<td>Image copies on tape</td>
<td>172</td>
</tr>
<tr>
<td>Termination of COPY</td>
<td>172</td>
</tr>
<tr>
<td>Restart of COPY</td>
<td>173</td>
</tr>
<tr>
<td>Sample COPY control statements</td>
<td>173</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 15. COPYTOCOPY</th>
<th>185</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax and options of the COPYTOCOPY control statement</td>
<td>186</td>
</tr>
<tr>
<td>Data sets that COPYTOCOPY uses</td>
<td>191</td>
</tr>
<tr>
<td>Concurrency and compatibility for COPYTOCOPY</td>
<td>193</td>
</tr>
<tr>
<td>Full or incremental image copies with COPYTOCOPY</td>
<td>194</td>
</tr>
<tr>
<td>Incremental image copies with COPYTOCOPY</td>
<td>194</td>
</tr>
<tr>
<td>Using more than one COPYTOCOPY statement</td>
<td>194</td>
</tr>
<tr>
<td>Copying from a specific image copy</td>
<td>195</td>
</tr>
<tr>
<td>Copying a FlashCopy image copy by using COPYTOCOPY</td>
<td>195</td>
</tr>
<tr>
<td>Using TEMPLATE with COPYTOCOPY</td>
<td>196</td>
</tr>
<tr>
<td>SYSCOPY records that are updated by COPYTOCOPY</td>
<td>196</td>
</tr>
<tr>
<td>How COPYTOCOPY determines which input copy to use</td>
<td>196</td>
</tr>
<tr>
<td>Generation data group definitions for the COPYTOCOPY utility</td>
<td>196</td>
</tr>
<tr>
<td>Using Db2 with DFSMS products</td>
<td>197</td>
</tr>
<tr>
<td>Image copies on tape</td>
<td>197</td>
</tr>
<tr>
<td>Copies of lists of objects from tape</td>
<td>197</td>
</tr>
<tr>
<td>Termination or restart of COPYTOCOPY</td>
<td>198</td>
</tr>
<tr>
<td>Sample COPYTOCOPY control statements</td>
<td>199</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 16. DIAGNOSE</th>
<th>203</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to force a utility abend</td>
<td>210</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 17. EXEC SQL</th>
<th>211</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Chapter 18. LISTDEF</th>
<th>215</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax and options of the LISTDEF control statement</td>
<td>215</td>
</tr>
<tr>
<td>Concurrency and compatibility for LISTDEF</td>
<td>225</td>
</tr>
<tr>
<td>Creating the LISTDEF control statement</td>
<td>226</td>
</tr>
</tbody>
</table>
Chapter 19. LOAD .................................................. 239
Syntax and options of the LOAD control statement ................................................................. 242
Before running LOAD .................................................................................................................. 305
Data sets that LOAD uses ............................................................................................................. 307
Concurrency and compatibility for LOAD ..................................................................................... 312
Preparing Db2 internal format input records that are not generated by UNLOAD for LOAD. .......... 314
When to use SORTKEYS NO ....................................................................................................... 314
Loading variable-length data ........................................................................................................ 315
How LOAD orders loaded records ............................................................................................... 315
Replacing data with LOAD ........................................................................................................... 315
Loading tables with special column types by using generated LOAD statements ....................... 318
Adding more data to a table or partition ....................................................................................... 320
Deleting all the data in a table space ............................................................................................ 320
Loading partitions ......................................................................................................................... 321
Loading partition-by-growth table spaces ..................................................................................... 324
Loading data containing XML columns ...................................................................................... 324
Loading delimited files .................................................................................................................. 325
Loading data with referential constraints .................................................................................... 328
Referential constraint violations .................................................................................................... 330
Compressing data by using the LOAD utility. .............................................................................. 330
How to load data from DL/I .......................................................................................................... 332
Loading data by using the cross-loader function. ........................................................................ 333
How to use inline COPY with LOAD ........................................................................................... 334
Creating a FlashCopy image copy with LOAD ............................................................................. 335
Improving LOAD performance ...................................................................................................... 336
Conversion of input data ............................................................................................................... 340
Specifying input fields ................................................................................................................... 342
Specifying the TRUNCATE and STRIP options ............................................................................ 342
How LOAD builds indexes while loading data .............................................................................. 342
Building indexes in parallel for LOAD .......................................................................................... 343
How LOAD leaves free space ........................................................................................................ 346
Loading with RECOVER-pending, REBUILD-pending, or REORG-pending status ....................... 346
Exit procedures. ............................................................................................................................. 347
Loading ROWID columns ............................................................................................................ 347
Loading a LOB column ................................................................................................................. 348
LOAD LOG on a LOB table space ................................................................................................. 349
Loading an XML column .............................................................................................................. 349
LOAD LOG on an XML table space ............................................................................................. 350
Running LOAD RESUME YES SHRLEVEL CHANGE without logging ........................................ 350
Collecting inline statistics while loading a table ............................................................................ 351
Inline COPY for a base table space ............................................................................................... 352
Termination of LOAD ................................................................................................................... 352
Restart of LOAD ........................................................................................................................... 353
After running LOAD ...................................................................................................................... 355
Copying the loaded table space or partition .................................................................................. 355
Resetting restricted status after running the LOAD utility ........................................................... 355
Running CHECK INDEX after loading a table that has indexes. .................................................... 359
Recovering data after a failed LOAD job ....................................................................................... 359
Reorganization of an auxiliary index after LOAD .......................................................................... 360
Effects of running LOAD ............................................................................................................. 360
Sample LOAD control statements. ............................................................................................... 362
Rebuilding multiple indexes

Chapter 20. MERGECOPY ................................................................. 377
Syntax and options of the MERGECOPY control statement .................. 378
Data sets that MERGECOPY uses ................................................. 380
Concurrency and compatibility for MERGECOPY .............................. 382
Full or incremental image copy ..................................................... 382
How MERGECOPY determines which input copy to use ...................... 383
Using MERGECOPY with individual data sets ................................. 383
Using MERGECOPY or COPY ...................................................... 384
Avoiding MERGECOPY LOG RBA inconsistencies .............................. 384
Termination or restart of MERGECOPY .......................................... 384
Sample MERGECOPY control statements ....................................... 385

Chapter 21. MODIFY RECOVERY ......................................................... 387
How MODIFY RECOVERY deletes rows ........................................... 393
Reclaiming space in the DBD ....................................................... 395
Improving REORG performance after adding a column ......................... 395
The effect of MODIFY RECOVERY on version numbers ..................... 396
Sample MODIFY RECOVERY control statements ............................... 397

Chapter 22. MODIFY STATISTICS .................................................... 399
Syntax and options of the MODIFY STATISTICS control statement ......... 400
Data sets that MODIFY STATISTICS uses ....................................... 402
Concurrency and compatibility for MODIFY STATISTICS ................. 402
Guidelines for deciding which statistics history rows to delete .............. 403
Deletion of specific statistics history rows ..................................... 403
Termination or restart of MODIFY STATISTICS ............................... 403
Sample MODIFY STATISTICS control statements ............................. 404

Chapter 23. OPTIONS ................................................................. 407
Syntax and options of the OPTIONS control statement ....................... 407
Concurrency and compatibility for OPTIONS ................................... 410
Executing statements in preview mode ......................................... 411
Specifying LISTDEF and TEMPLATE libraries ................................. 411
Overriding standard utility processing behavior ............................... 411
Termination or restart of OPTIONS .............................................. 412
Sample OPTIONS control statements .......................................... 412

Chapter 24. QUIESCE .............................................................. 415
Syntax and options of the QUIESCE control statement ....................... 416
Before running QUIESCE .......................................................... 418
Data sets that QUIESCE uses ..................................................... 418
Concurrency and compatibility for QUIESCE ................................ 419
Use of QUIESCE on catalog and directory objects ............................ 420
Common quiesce points ............................................................ 421
Running QUIESCE on a table space in pending status ....................... 422
Reasons why QUIESCE fails to write to disk .................................. 422
Termination and restart of QUIESCE .......................................... 422
Sample QUIESCE control statements ........................................... 423

Chapter 25. REBUILD INDEX ......................................................... 427
Syntax and options of the REBUILD INDEX control statement .......... 428
Before running REBUILD INDEX .................................................. 440
Data sets that REBUILD INDEX uses ......................................... 440
Concurrency and compatibility for REBUILD INDEX ......................... 443
Access with REBUILD INDEX SHRLEVEL ..................................... 445
Rebuilding index partitions ......................................................... 445
Rebuilding indexes on partition-by-growth table spaces ....................... 446
How to improve performance when rebuilding index partitions .......... 446
Rebuilding multiple indexes .......................................................... 447
Chapter 26. RECOVER ................................................................. 459
Syntax and options of the RECOVER control statement .................. 461
Before running RECOVER .......................................................... 472
  Data sets that RECOVER uses ................................................. 472
  Concurrency and compatibility for RECOVER .......................... 473
Recovering with a system-level backup .................................... 475
How to determine which system-level backups Db2 recovers .............. 476
Determining which recovery base Db2 uses ................................ 477
Determining whether the system-level backups reside on disk or tape ... 477
Recovering a table space .......................................................... 478
Recovering a list of objects ....................................................... 479
Recovering a data set or partition .............................................. 480
Recovery with incremental copies .............................................. 480
Recovering with FlashCopy image copies ................................... 481
Recovering a page .................................................................... 482
Recovering an error range ........................................................ 483
Effect on RECOVER of the NOT LOGGED or LOGGED table space attributes . 484
Recovering with a data set copy that is not made by Db2 ....................... 484
Recovering catalog and directory objects ..................................... 485
  Objects that contain recovery information ................................ 490
    Point-in-time recovery of the catalog, directory, and all user objects 492
Reinitializing DSNDDB01.SYSUTILX ............................................. 493
Recovering a table space that contains LOB or XML data .................. 494
Recovering a table space that contains clone objects ....................... 495
Point-in-time recovery ............................................................. 495
Avoiding specific image copy data sets during a recovery .......... 504
How to improve RECOVER performance .................................... 505
Optimizing the LOGAPPLY phase .............................................. 506
Recovering image copies in a JES3 environment ............................. 507
Resetting RECOVER-pending or REBUILD-pending status ............ 507
How the RECOVER utility performs fallback recovery ..................... 508
How the RECOVER utility retains tape mounts .............................. 509
Avoiding damaged media .......................................................... 509
Termination or restart of RECOVER ........................................... 509
Effects of running RECOVER ..................................................... 510
Sample RECOVER control statements ......................................... 511

Chapter 27. REORG INDEX .......................................................... 515
Syntax and options of the REORG INDEX control statement ............. 516
Before running REORG INDEX .................................................... 533
  Data sets that REORG INDEX uses ........................................... 535
  Concurrency and compatibility for REORG INDEX .................... 539
Determining which indexes require reorganization ....................... 541
Using the LEAFDISTLIMIT and REPORTONLY options to determine when reorganization is needed 542
Access with REORG INDEX SHRLEVEL ........................................ 542
Creating a FlashCopy image copy with REORG INDEX .................... 544
Temporarily interrupting REORG ............................................... 545
Improving performance with REORG INDEX .................................. 545
Termination of REORG INDEX .................................................... 546
Restart of REORG INDEX .......................................................... 547
Review of REORG INDEX output ................................................. 548
Effect of REORG INDEX on index version numbers ....................... 549
<table>
<thead>
<tr>
<th>Data type conversion with the UNLOAD utility</th>
<th>870</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output field types</td>
<td>871</td>
</tr>
<tr>
<td>Output field positioning and size</td>
<td>873</td>
</tr>
<tr>
<td>Layout of output fields</td>
<td>873</td>
</tr>
<tr>
<td>Output for special values Infinity, sNaN, or NaN</td>
<td>876</td>
</tr>
<tr>
<td>Unloading delimited files</td>
<td>876</td>
</tr>
<tr>
<td>Specifying TRUNCATE and STRIP options for output data</td>
<td>879</td>
</tr>
<tr>
<td>LOAD statements that are generated by UNLOAD</td>
<td>880</td>
</tr>
<tr>
<td>Unloading compressed data</td>
<td>881</td>
</tr>
<tr>
<td>Field specification errors</td>
<td>881</td>
</tr>
<tr>
<td>Sample UNLOAD control statements</td>
<td>881</td>
</tr>
</tbody>
</table>

**Part 3. Db2 stand-alone utilities** ........................................... 889

**Chapter 36. Invoking stand-alone utilities** .................................. 891
Specifying options for stand-alone utilities by using the JCL EXEC PARM parameter | 892 |
Stand-alone utility control statements ........................................... 892

**Chapter 37. DSNJCNVB** ............................................................... 895

**Chapter 38. DSNJCNVT** ............................................................... 897

**Chapter 39. DSNJLOGF (preformat active log)** ................................ 901

**Chapter 40. DSNJU003 (change log inventory)** ................................ 903
Making changes for active logs ....................................................... 918
Making changes for archive logs ..................................................... 920
A conditional restart control record ............................................. 921
Deleting log data sets with errors ............................................... 921
Altering references to log data sets in the BSDS ............................ 922
Defining the high-level qualifier for catalog and directory objects .......... 923
Renaming Db2 system data sets ...................................................... 923
Renaming Db2 active log data sets ................................................ 924
Renaming Db2 archive log data sets .............................................. 924
Sample DSNJU003 control statements ........................................... 924

**Chapter 41. DSNJU004 (print log map)** ......................................... 927
Syntax and options of the DSNJU004 control statement ......................... 928
Sample DSNJU004 control statement ............................................... 929
DSNJU004 (print log map) output .................................................. 929

**Chapter 42. DSNJU008 (print CDDS)** ........................................... 945
Syntax and options of the DSNJU008 control statement ......................... 945
DSNJU008 examples ................................................................. 948

**Chapter 43. DSN1COMP** ............................................................... 949
Syntax and options of the DSN1COMP control statement ......................... 951
Before running DSN1COMP .......................................................... 954
How to estimate compression savings achieved with option REORG .......... 955
Free space in compression calculations on table space ......................... 955
Sample DSN1COMP control statements ........................................... 956
DSN1COMP output .......................................................................... 958

**Chapter 44. DSN1COPY** ............................................................... 961
Syntax and options of the DSN1COPY control statement ......................... 964
Before running DSN1COPY ............................................................ 970
   Data sets that DSN1COPY uses .................................................. 972
Inconsistent data checks .............................................................. 977

Contents xi
The effects of not specifying the OBIDXLAT option ........................................ 977
Requirements for using an image copy as input to DSN1COPY ....................... 978
Copying from an image copy ........................................................................... 978
Restoring indexes with DSN1COPY ................................................................. 979
Restoring table spaces with DSN1COPY ......................................................... 980
Printing with DSN1COPY .............................................................................. 981
Copying tables from one subsystem to another .............................................. 981
Sample DSN1COPY control statements ......................................................... 983

Chapter 45. DSN1LOGP .............................................................................. 989
Syntax and options of the DSN1LOGP control statement ............................... 991
Determining the PSID for base and clone objects ......................................... 998
Archive log data sets on tape .......................................................................... 998
Sample DSN1LOGP control statements ......................................................... 999
DSN1LOGP output .......................................................................................... 1002

Chapter 46. DSN1PRNT ............................................................................. 1009
Syntax and options of the DSN1PRNT control statement .............................. 1010
Printing with DSN1PRNT instead of DSN1COPY ......................................... 1016
Determining the page size and data set size for DSN1PRNT ......................... 1016
Sample DSN1PRNT control statements .......................................................... 1017

Chapter 47. DSN1SDMP ............................................................................ 1021
Syntax and options of the DSN1SDMP control statement ............................ 1022
Assigning buffers ......................................................................................... 1027
Conditions for generating a dump ................................................................. 1028
Stopping or modifying DSN1SDMP traces .................................................... 1028
Sample DSN1SDMP control statements ....................................................... 1029

Part 4. Appendixes ..................................................................................... 1033

Appendix A. Supplied stored procedures for utility operations ....................... 1035

Appendix B. Advisory or restrictive states ...................................................... 1037
Auxiliary CHECK-pending status .................................................................. 1037
Auxiliary warning status .............................................................................. 1038
CHECK-pending status .................................................................................. 1039
COPY-pending status .................................................................................... 1040
DBETE status ............................................................................................... 1041
Group buffer pool RECOVER-pending status ................................................. 1042
Informational COPY-pending status ............................................................... 1042
PRO restricted status ..................................................................................... 1043
REBUILD-pending status .............................................................................. 1043
RECOVER-pending status .............................................................................. 1044
REFRESH-pending status .............................................................................. 1045
REORG-pending status .................................................................................. 1045
Restart-pending status .................................................................................. 1048

Appendix C. DSN1SMFP .......................................................................... 1051
Before running DSN1SMFP .......................................................................... 1053
Running DSN1SMFP ...................................................................................... 1053
Sample DSN1SMFP control statement ......................................................... 1054
DSN1SMFP output ......................................................................................... 1054
DSN1SMFP output record structure .............................................................. 1056
Standard report header .................................................................................. 1056
IFCID003: DDF Location .............................................................................. 1057
IFCID004: Trace Start .................................................................................... 1059
IFCID005: Trace Stop .................................................................................... 1059
IFCID023: Utility Start .................................................................................. 1059
Appendix D. DSNADMSB ........................................ 1099
Parameters of the DSNADMSB program .......................... 1100
Before running DSNADMSB .................................... 1108
Data sets that DSNADMSB uses ................................ 1109
Copying the data that DSNADMSB and ADMIN_INFO_SQL collect to another subsystem ............... 1109
Examples of DSNADMSB invocation .............................. 1110

Appendix E. DSNTSMFD ......................................... 1119
Before running DSNTSMFD ....................................... 1119
Data sets that DSNTSMFD uses ................................ 1120
Examples of DSNTSMFD invocation .............................. 1120

Appendix F. How real-time statistics are used by Db2 utilities ....... 1123

Appendix G. Delimited file format .............................. 1125
Data types in delimited files .................................... 1126
Examples of delimited files .................................... 1127

Information resources for Db2 11 for z/OS and related products .......... 1129

Notices ..................................................................... 1131
Programming interface information .................................. 1132
Trademarks ............................................................ 1133
Terms and conditions for product documentation ................. 1134
Privacy policy considerations ...................................... 1134

Glossary .................................................................... 1137

Index ...................................................................... 1139
About this information

This information contains usage information for the tasks of system administration, database administration, and operation. It presents detailed information about using utilities, specifying syntax (including keyword and parameter descriptions), and starting, stopping, and restarting utilities. This book also includes job control language (JCL) and control statements for each utility.

Throughout this information, “Db2®” means “Db2 11 for z/OS®”. References to other Db2 products use complete names or specific abbreviations.

**Important:** To find the most up to date content, always use [IBM® Knowledge Center](https://www.ibm.com/knowledgecenter) which is continually updated as soon as changes are ready. PDF manuals are updated only when new editions are published, on an infrequent basis.

This information assumes that Db2 11 is running in new-function mode, and that your application is running with the application compatibility value of 'V11R1'.

### Availability of new function in Db2 11

The behavior of data definition statements such as CREATE, ALTER, and DROP, which embed data manipulation SQL statements that contain new capabilities, depends on the application compatibility value that is in effect for the application. An application compatibility value of 'V11R1' must be in effect for applications to use new capability in embedded statements such as SELECT, INSERT, UPDATE, DELETE, MERGE, CALL, and SET assignment-statement. Otherwise, an application compatibility value of 'V10R1' can be used for data definition statements.

Generally, new SQL capabilities, including changes to existing language elements, functions, data manipulation statements, and limits, are available only in new-function mode with applications set to an application compatibility value of 'V11R1'.

Optimization and virtual storage enhancements are available in conversion mode unless stated otherwise.

SQL statements can continue to run with the same expected behavior as in DB2® 10 new-function mode with an application compatibility value of 'V10R1'.

Who should read this information

This information is intended for system administrators, database administrators, system operators, and application programmers of Db2 online and stand-alone utilities.

**Recommendation:** Familiarize yourself with Db2 for z/OS prior to using this book.

### Db2 Utilities Suite for z/OS

**Important:** In Db2 11, the Db2 Utilities Suite for z/OS is available as an optional product. You must separately order and purchase a license to such utilities, and discussion of those utility functions in this publication is not intended to otherwise imply that you have a license to them.
Db2 utilities can use the DFSORT program regardless of whether you purchased a license for DFSORT on your system. For more information, see the following informational APARs:

- II14047
- II14213
- II13495

Db2 utilities can use IBM Db2 Sort for z/OS (5655-W42) as an alternative to DFSORT for utility SORT and MERGE functions. Use of Db2 Sort for z/OS requires the purchase of a Db2 Sort for z/OS license. For more information about Db2 Sort for z/OS, see [Db2 Sort for z/OS](https://www.ibm.com/training/)

**Related information:**

### Terminology and citations

When referring to a Db2 product other than Db2 for z/OS, this information uses the product's full name to avoid ambiguity.

**About the Db2 brand change:** IBM is rebranding DB2 to Db2. As such, there will be changes to all the Db2 offerings. For example, “DB2 for z/OS” is now referred to as “Db2 for z/OS,” beginning with Db2 11. While IBM implements the change across the Db2 family of products, you might see references to the original name “DB2 for z/OS” or “DB2” in different IBM web pages and documents. “Db2 for z/OS” and “Db2 for z/OS” refer to the same product, when the PID, Entitlement, version, modification, and release information match. For more information, see [Revised naming for IBM Db2 family products](https://www.ibm.com/support/docview.wss?uid=swg21388489).

The following terms are used as indicated:

**Db2**

- Represents either the Db2 licensed program or a particular Db2 subsystem.

**Tivoli® OMEGAMON® XE for Db2 Performance Expert on z/OS**

- Refers to any of the following products:
  - IBM Tivoli OMEGAMON XE for Db2 Performance Expert on z/OS
  - IBM Db2 Performance Monitor on z/OS
  - IBM Db2 Performance Expert for Multiplatforms and Workgroups
  - IBM Db2 Buffer Pool Analyzer for z/OS

**C, C++, and C language**

- Represent the C or C++ programming language.

**CICS®**

- Represents CICS Transaction Server for z/OS.

**IMS™**

- Represents the IMS Database Manager or IMS Transaction Manager.

**MVS™**

- Represents the MVS element of the z/OS operating system, which is equivalent to the Base Control Program (BCP) component of the z/OS operating system.

**RACF®**

- Represents the functions that are provided by the RACF component of the z/OS Security Server.

### Accessibility features for Db2 11 for z/OS

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

The following list includes the major accessibility features in z/OS products, including Db2 11 for z/OS. These features support:

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

**Tip:** The IBM Knowledge Center (which includes information for Db2 for z/OS) and its related publications are accessibility-enabled for the IBM Home Page Reader. You can operate all features using the keyboard instead of the mouse.

**Keyboard navigation**

For information about navigating the Db2 for z/OS ISPF panels using TSO/E or ISPF, refer to the z/OS TSO/E Primer, the z/OS TSO/E User’s Guide, and the z/OS ISPF User’s Guide. These guides describe how to navigate each interface, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

**Related accessibility information**

**IBM and accessibility**

See the IBM Accessibility Center at [http://www.ibm.com/able](http://www.ibm.com/able) for more information about the commitment that IBM has to accessibility.

**How to send your comments**

Your feedback helps IBM to provide quality information. Please send any comments that you have about this book or other Db2 for z/OS documentation.

Send your comments by email to [db2zinfo@us.ibm.com](mailto:db2zinfo@us.ibm.com) and include the name of the product, the version number of the product, and the number of the book. If you are commenting on specific text, please list the location of the text (for example, a chapter and section title or a help topic title).

**Naming conventions used in this information**

When you use Db2 commands and utilities, be aware of the applicable naming conventions.

When you use a parameter for an object that is created by SQL statements (for example, tables, table spaces, and indexes), identify the object by following the SQL syntactical naming conventions.

In this information, characters are classified as **letters, digits, or special characters**.

- A **letter** is any one of the uppercase characters A through Z (including the three characters that are reserved in the United States as alphabetic extenders for national languages, #, @, and $).
- A **digit** is any one of the characters 0 through 9.
- A **special character** is any character other than a letter or a digit.
**authorization-id**
A short identifier of one to eight letters, digits, or the underscore that identifies a set of privileges. An authorization ID must begin with a letter.

**connection-name**
An identifier of one to eight characters that identifies an address space connection to Db2. A connection identifier is one of the following values:
- TSO (for DSN processes that run in TSO foreground).
- BATCH (for DSN processes that run in TSO batch).
- DB2CALL (for the call attachment facility (CAF)).
- The system identification name (for IMS and CICS processes).

**Related information:**
Managing connection requests from local applications (Managing Security)

**correlation-id**
An identifier of 1 to 12 characters that identifies a process within an address space connection. A correlation ID must begin with a letter.

A correlation ID can be one of the following values:
- The TSO logon identifier (for DSN processes that run in TSO foreground and for CAF processes).
- The job name (for DSN processes that run in TSO batch).
- The PST#.PSBNAME (for IMS processes).
- The entry identifier.thread_number.transaction_identifier (for CICS processes).

**cursor-name**
An identifier that designates a result set. Cursor names that are specified with the EXEC SQL and LOAD utilities cannot be longer than eight characters.

**database-name**
A short identifier that identifies a database. The identifier must start with a letter and must not include special characters.

**data-set-name**
An identifier of 1 to 44 characters that identifies a data set.

**dbrm-member-name**
An identifier of one to eight letters or digits that identifies a member of a partitioned data set.

A DBRM member name should not begin with DSN because of a potential conflict with Db2-provided DBRM member names. If you specify a DBRM member name that begins with DSN, Db2 issues a warning message.

**dbrm-pds-name**
An identifier of 1 to 44 characters that identifies a partitioned data set.

**ddname**
An identifier of one to eight characters that identifies the name of a DD statement.

**hexadecimal-constant**
A sequence of digits or any of the letters from A to F (uppercase or lowercase).

**hexadecimal-string**
An X followed by a sequence of characters that begins and ends with the string delimiter, an apostrophe. The characters between the string delimiters must be a hexadecimal number.
index-name
A qualified or unqualified name that identifies an index.

A qualified index name is a schema name followed by a period and an identifier.

An unqualified index name is an identifier with an implicit schema name qualifier. The implicit schema is determined by the SQL rules for unqualified types, functions, procedures, global variables, and specific names.

If the index name contains a blank character, the name must be enclosed in quotation marks when specified in a utility control statement.

Related information:
Unqualified type, function, procedure, global variable, and specific names (Db2 SQL)

location-name
A location identifier of 1 to 16 letters (but excluding the alphabetic extenders),
digits, or the underscore that identifies an instance of a database management system. A location name must begin with a letter.

luname
An SQL short identifier of one to eight characters that identifies a logical unit name. A LU name must begin with a letter.

member-name
An identifier of one to eight letters (including the three alphabetic extenders) or digits that identifies a member of a partitioned data set.

A member name should not begin with DSN because of a potential conflict with Db2-provided member names. If you specify a member name that begins with DSN, Db2 issues a warning message.

qualifier-name
An SQL short identifier of one to eight letters, digits, or the underscore that identifies the implicit qualifier for unqualified table names, views, indexes, and aliases.

string
A sequence of characters that begins and ends with an apostrophe.

subsystem-name
An identifier that specifies the Db2 subsystem as it is known to the operating system.

table-name
A qualified or unqualified name that designates a table.

A fully qualified table name is a three-part name. The first part is a location name that designates the DBMS at which the table is stored. The second part is a schema name. The third part is an SQL identifier. A period must separate each of the parts.

A two-part table name is implicitly qualified by the location name of the current server. The first part is a schema name. The second part is an SQL identifier. A period must separate the two parts.

A one-part or unqualified table name is an SQL identifier with two implicit qualifiers. The first implicit qualifier is the location name of the current server. The second is a schema name, which is determined by the SQL rules for unqualified types, functions, procedures, global variables, and specific names.
If the table name contains a blank, the name must be enclosed in quotation marks when specified in a utility control statement.

**Related information:**

Unqualified type, function, procedure, global variable, and specific names (Db2 SQL)

**table-space-name**

A short identifier that identifies a table space of an identified database. The identifier must start with a letter and must not include special characters. If a database is not identified, a table space name specifies a table space of database DSNDDB04.

**utility-id**

An identifier of 1 to 16 characters that uniquely identifies a utility process within Db2. A utility ID must begin with a letter. The remaining characters can be uppercase and lowercase letters, numbers 0 through 9, and the following characters: #, $, ., ¬, and @.

**Related concepts:**

Naming conventions (Db2 SQL)

SQL identifiers (Db2 SQL)

---

**How to read syntax diagrams**

Certain conventions apply to the syntax diagrams that are used in IBM documentation.

Apply the following rules when reading the syntax diagrams that are used in Db2 for z/OS documentation:

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line.
  
  The ►► symbol indicates the beginning of a statement.
  
  The ──► symbol indicates that the statement syntax is continued on the next line.
  
  The ►── symbol indicates that a statement is continued from the previous line.
  
  The ──► symbol indicates the end of a statement.

- Required items appear on the horizontal line (the main path).

  ►►required_item◄◄

- Optional items appear below the main path.

  ►►required_item◄◄optional_item◄◄

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

  ►►required_item◄◄optional_item◄◄

- If you can choose from two or more items, they appear vertically, in a stack.
If you must choose one of the items, one item of the stack appears on the main path.

►►required_item►►required_choice1►►required_choice2

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

►►required_item►►optional_choice1►►optional_choice2

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

►►required_item►►default_choice►►optional_choice►►optional_choice

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

►►required_item►►repeatable_item

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

►►required_item►►repeatable_item,

A repeat arrow above a stack indicates that you can repeat the items in the stack.

- Sometimes a diagram must be split into fragments. The syntax fragment is shown separately from the main syntax diagram, but the contents of the fragment should be read as if they are on the main path of the diagram.

►►required_item►►fragment-name

fragment-name:

►►required_item◄►optional_name

- With the exception of XPath keywords, keywords appear in uppercase (for example, FROM). Keywords must be spelled exactly as shown. XPath keywords are defined as lowercase names, and must be spelled exactly as shown. Variables appear in all lowercase letters (for example, column-name). They represent user-supplied names or values.
• If punctuation marks, parentheses, arithmetic operators, or other such symbols are shown, you must enter them as part of the syntax.

Related information:
Part 1. Basic information about the Db2 utilities

Use IBM Db2 utilities to help maintain data in your Db2 for z/OS databases.

**Syntax diagrams for utilities**

For information about the conventions for syntax diagrams in IBM documentation, see .

**Online and stand-alone utilities**

Most of the basic utilities, such as LOAD and COPY, are online utilities; they require Db2 to be up and running. Db2 online utilities run as standard batch jobs or stored procedures. They do not run under control of the terminal monitor program (TMP); they have their own attachment mechanism and they invoke Db2 control facility services directly. For more information about online utilities, see [Part 2, “Db2 online utilities,” on page 13](#) and [Chapter 6, “Invoking Db2 online utilities,” on page 15](#).

The other type of utilities is stand-alone utilities. These utilities run independently of Db2. They work directly on the data sets. You can use stand-alone utilities to do tasks such as format logs, modify the bootstrap data set (BSDS), and copy and print data sets. The stand-alone utilities run as batch jobs. The only way to run these utilities is to use JCL. For more information about stand-alone utilities, see [Part 3, “Db2 stand-alone utilities,” on page 889](#) and .

**Exception:** Some Db2 for z/OS users use the term online utilities to mean something other than what IBM means when it uses this term in the documentation. Some users use this term to refer to utilities that leave the data available while they are running. For example, when you use the REORG TABLESPACE utility to reorganize data, you can specify the level of access that other applications and processes have to that same data. If you specify the SHRLEVEL CHANGE option, the data can be changed while the utility runs. Therefore, REORG TABLESPACE SHRLEVEL CHANGE is considered by some users to be an ‘online utility’ because the data is never taken offline. However, IBM does not use the term this way.

**Related concepts:**

- [Db2 utilities (Introduction to Db2 for z/OS)](#)

**Related information:**

- [Db2 for z/OS Utilities in Practice](#)
Chapter 1. Db2 utilities packaging

Several utilities are included with Db2 at no extra charge. Other utilities are available in the separate Db2 Utilities Suite for z/OS product.

The following utilities are core utilities, which are included (at no extra charge) with Db2 11 for z/OS:
- CATENFM
- CATMAINT
- DIAGNOSE
- LISTDEF
- OPTIONS
- QUIESCE
- REPAIR
- REPORT
- TEMPLATE
- All DSN stand-alone utilities

All other utilities are included in the Db2 Utilities Suite for z/OS (5655-W87, FMID JDABB1K):
- BACKUP SYSTEM
- CHECK DATA
- CHECK INDEX
- CHECK LOB
- COPY
- COPYTOCOPY
- EXEC SQL
- LOAD
- MERGECOPY
- MODIFY RECOVERY
- MODIFY STATISTICS
- REBUILD INDEX
- RECOVER
- REORG INDEX
- REORG TABLESPACE
- RESTORE SYSTEM
- RUNSTATS
- STOSPACE
- UNLOAD

All Db2 utilities operate on catalog, directory, and sample objects, without requiring purchase of the Db2 Utilities Suite for z/OS.

SMP/E jobs for Db2 utility products

To load the Db2 utility products, use System Modification Program Extended (SMP/E). SMP/E processes the installation cartridges and creates Db2 distribution target libraries.

Db2 provides several jobs that invoke SMP/E. These jobs are on the cartridge that you received with the utility product:
SMP/E RECEIVE job, DSNRECVK
Job DSNRECVK loads the Db2 Utilities Suite for z/OS Db2 11 program modules, macros, and procedures into temporary data sets (SMPTLIBs).

SMP/E APPLY job, DSNAPPLK
Job DSNAPPLK, copies and link-edits the program modules, macros, and procedures for the Db2 Utilities Suite for z/OS Db2 11 into the Db2 target libraries.

SMP/E ACCEPT job, DSNACCPK
Job DSNACCPK, copies the program modules, macros, and procedures for the Db2 Utilities Suite Db2 11 into the Db2 distributed libraries.

The prologue for each job contains directions for tailoring the job for your site. Follow the instructions carefully to ensure that your SMP/E process for Db2 Utilities Suite for z/OS works correctly.

For detailed instructions for installing the Db2 Utilities Suite for z/OS product, see [Db2 Utilities Suite for z/OS Program Directory](#).

**Operation of Db2 utilities in a mixed-release data sharing environment**

The utilities batch module, DSNUTILB, is split into multiple parts:
- a release-independent module called DSNUTILB
- a release-dependent module DSNUT111
- utility-dependent load modules

To operate in a mixed-release data sharing environment, you must have the following modules available to the utility jobs that operate across the data sharing group:
- The release-dependent modules from both releases
- All applicable utility-dependent modules

**Related concepts:**
[System installation and maintenance using SMP/E (z/OS basic skills)](#)

**Related reference:**
[Load module names for running purchased utilities in coexistence (Db2 Installation and Migration)](#)

**Related information:**
Chapter 2. Privileges and authorization IDs for Db2 utilities

A utility job can be issued by an individual user, a program that runs in batch mode, or an IMS or CICS transaction. The term process describes any of these initiators.

Db2 processes are represented by a set of identifiers (IDs), which are called authorization IDs. What the process can do with Db2 is determined by the privileges and authorities that are held by its identifiers.

For Db2 online utilities, the process can be represented by the following identifiers:

• A primary authorization ID.
• Possibly one or more secondary IDs.
• A role, if the process is running in a trusted connection with an associated role.

For example, a process can have a secondary authorization ID that is a Resource Access Control Facility (RACF) group ID. Suppose that a RACF group holds the LOAD privilege on a particular database. Any member of the group can run the LOAD utility to load table spaces in that database.

The privileges that are required for each utility are listed in the documentation for the utility.

Required authorizations for invoking utilities on tables that have multilevel security with row-level granularity

If you use RACF access control with multilevel security, you need additional authorizations to run the following utilities on tables that have multilevel security with row-level granularity:

• LOAD
• UNLOAD
• REORG TABLESPACE

The authorization requirements are listed in the documentation for each of these utilities.

All other utilities, including all stand-alone utilities, ignore the row-level granularity. They check only for authorization to operate on the table space; they do not check row-level authorization.

Db2 online utilities in a trusted connection

Db2 online utilities can run in a trusted connection if both of the following conditions are true:

• A matching trusted context is defined where the primary authorization ID matches the trusted context SYSTEM AUTHID.
• The job name matches the JOBNAME attribute that is defined for the identified trusted context.

The primary authorization ID can acquire a special set of privileges in a trusted context, by roles.
Related concepts:
- Authorization IDs (Managing Security)
- Privileges and authorities (Managing Security)
- Multilevel security (Managing Security)

Related information:
- Managing access through trusted contexts (Managing Security)
Chapter 3. Utilities that can be run on declared temporary objects

You can run the REPAIR and STOSPACE utilities on declared temporary objects.

You can use REPAIR with the DBD option on declared temporary tables. Declared temporary tables must be created in a database that is defined with the AS TEMP clause.

You can use STOSPACE on storage groups that have objects within temporary databases.

No other Db2 utilities can be used on a declared temporary table, its indexes, or its table spaces.

Related concepts:
- Creation of temporary tables (Introduction to Db2 for z/OS)

Related tasks:
- “Repairing DBDs” on page 696

Related reference:
- Chapter 33, “STOSPACE,” on page 787
Chapter 4. Effect of utilities on objects that have the DEFINE NO attribute

When you create a table space or index space with the DEFINE NO attribute, the object is defined; however, Db2 does not allocate the associated data sets until a row is inserted or loaded into a table in that table space.

You can populate table spaces whose data sets are not yet defined by using the LOAD utility with either the RESUME option or the REPLACE option. In this case, LOAD allocates the data sets, updates the SPACE column in the catalog table to indicate that data sets exist, and loads the specified table space. For a partitioned table space, all partitions are allocated even if the utility loads only one partition. Do not attempt to populate a partitioned table space with concurrent LOAD PART jobs until after one job creates all of the data sets.

If one of the following online utilities encounters a target object with the DEFINE NO attribute, the utility issues informational message DSNU185I, skips that object, and continues processing:

- CHECK DATA
- CHECK INDEX
- COPY
- MERGECOPY
- MODIFY RECOVERY
- QUIESCE
- REBUILD INDEX
- RECOVER
- REORG INDEX
- REORG TABLESPACE (One exception is if ROWFORMAT RRF is specified in the REORG statement and the specified target is an undefined table space in basic row format. In this case, REORG updates the row format definition in the catalog and directory. No data sets are defined for the table space.)
- REPAIR, but not REPAIR DBD
- RUNSTATS TABLESPACE INDEX(ALL)
- RUNSTATS INDEX ()
- UNLOAD

RUNSTATS recognizes DEFINE NO objects and updates the access path statistics in the Db2 catalog to reflect the empty objects.

The SYSIBM.SYSSSTATFEEDBACK table might contain rows that recommend collection of statistics for DEFINE NO objects or empty table space partitions. However, statistics cannot be collected for such objects until after data is inserted and the underlying data sets are created.

You cannot use stand-alone utilities on objects whose data sets are not yet defined.

Related tasks:
- Deferring allocation of Db2-managed data sets (Db2 Administration Guide)

Related information:
DSNU185I (Db2 Messages)
Chapter 5. Effect of utilities on encrypted data

In Db2, you can encrypt data through built-in functions. You can copy and recover this encrypted data. You can also move encrypted data between systems. Data remains encrypted throughout these processes.

Running any of the following utilities on encrypted data might produce unexpected results:
- CHECK DATA
- LOAD
- REBUILD INDEX
- REORG TABLESPACE
- REPAIR
- RUNSTATS
- UNLOAD
- DSN1PRNT

Related concepts:

Encrypting your data through Db2 built-in functions (Managing Security)
Part 2. Db2 online utilities

Db2 online utilities run as standard batch jobs or stored procedures, and they require Db2 to be running. They do not run under control of the terminal monitor program (TMP); they have their own attachment mechanism and they invoke Db2 control facility services directly.
Chapter 6. Invoking Db2 online utilities

To invoke Db2 online utilities, Db2 must be up and running.

Before you begin

Make sure that you are authorized to run the utility that you want to invoke. You can check the required authorization in the documentation for each utility.

Procedure

To run Db2 online utilities:

1. Prepare the necessary data sets.
   For more information about allocating data sets for utilities, see “Data sets that online utilities use” on page 16. For a list of the data sets that are required for a utility, see the documentation for that utility.

2. Create a utility control statement.
   For general information about utility control statements, see “Utility control statements” on page 18. For information about statement syntax for a utility, see the documentation for that utility.

   Some utilities can operate at the partition level. For those utilities, any reference in the utility statement to a partition number denotes the physical partition number unless otherwise specified. Similarly, any partition numbers that are included in the utility output are the physical partition numbers.

3. Check for any concurrency and compatibility restrictions. These restrictions limit what other utilities and processes can run at the same time. For a list of restrictions, see the documentation for the utility that you want to use.

   If the utility supports parallelism, it can use additional threads to support the parallel subtasking. Consider increasing the values of subsystem parameters that control threads, such as MAX BATCH CONNECT and MAX USERS.

4. Invoke the utility by using one of the following methods:

   Requirement: In the JCL for all utility jobs, specify a load library (prefix.SDSNLOAD) that is at a maintenance level that is compatible with the Db2 subsystem. Otherwise, errors can occur.

   - “Invoking an online utility by using the DB2 Utilities panel in DB2I” on page 21
   - “Invoking a Db2 utility by using the DSNU CLIST command in TSO” on page 26
   - “The supplied JCL procedure (DSNUPROC) for invoking a Db2 online utility” on page 33
   - “Invoking a Db2 online utility by creating the JCL data set yourself” on page 36

   - Invoke utilities from a Db2 application program by calling stored procedure DSNUTILU stored procedure (Db2 SQL)

Data Sharing: When you submit a utility job, specify the name of the Db2 subsystem to which the utility is to attach, the group attachment name, or subgroup attachment name. If you do not use the group attachment name or subgroup attachment name, the utility job must run on the z/OS system where
the specified Db2 subsystem is running. Ensure that the utility job runs on the appropriate z/OS system. To do so, use one of several z/OS installation-specific statements, such as:

- For JES2 multi-access spool (MAS) systems, insert the following statement into the utility JCL:
  
```*/JOBPARM SYSAFF=cccc```

- For JES3 systems, insert the following statement into the utility JCL:
  
```*/MAIN SYSTEM=main-name```  

Your installation might have other mechanisms for controlling where batch jobs run, such as using job classes.

**Results**

If the utility fails, you can either terminate it or restart it.

**Related concepts:**

- Chapter 2, “Privileges and authorization IDs for Db2 utilities,” on page 5
- Group attachment names and subgroup attachment names (Db2 Installation and Migration)

**Related tasks:**

- “Terminating an online utility” on page 43
- “Restarting an online utility” on page 44

---

### Data sets that online utilities use

Every online utility requires an input data set with the utility control statement. Some utilities require additional data sets.

In the utility job, use the SYSIN DD statement to define an in-stream data set with the utility statement. Use other DD statements to define any other data sets that the utility needs. Alternatively, you can use the TEMPLATE utility to define those data sets.

See the description of each utility for a list of data sets that it uses.

### General rules for utility data sets

Data sets for utilities have the following requirements and restrictions:

**For input data sets**

Db2 utilities use the logical record length (LRECL), the record format (RECFM) and the block size (BLKSIZE) with which the data set was created.

Variable-spanned (VS) or variable-blocked-spanned (VBS) record formats are not allowed for utility input data sets. The only exceptions are for the LOAD and UNLOAD utilities. These utilities use VBS data sets when you specify the FORMAT SPANNED YES option. Also, LOAD accepts VBS data sets when you specify the FORMAT SQL/DS option.

**For output data sets**

Db2 utilities determine both the logical record length (LRECL) and the record format (RECFM). Any specified values for LRECL or RECFM are ignored.
If you supply block size (BLKSIZE), that size is used; otherwise, the system
determines the optimal block size for the storage device. Db2 supports the
large block interface (LBI), which allows block sizes that are greater than
32 KB on certain tape drives. UNLOAD SPANNED YES ignores LBI for
output data sets. The TAPEBLKSZLIM parameter of the DEVSUPxx
member of SYS1.PARMLIB controls the block size limit for tapes.

Partitioned data sets (PDS) are not allowed for output data sets.

For output data sets for FlashCopy® image copies
The output VSAM data sets are allocated during the processing of the
DFSMSdss COPY command. The output data sets for FlashCopy image
copies are always cataloged.

For both input and output data sets
Db2 utilities use the value that you supply for the number of buffers
(BUFNO), with a maximum of 99 buffers. The default number of buffers is
20. The utilities set the number of channel programs equal to the number
of buffers. Utilities ignore the parameters that specify the buffer size
(BUFSIZE) and the number of channel programs (NCP).

Increasing the number of buffers (BUFNO) can result in an increase in real
storage usage and page fixing below the 16-MB line.

If you omit any data control block (DCB) parameters, the utilities use
default values.

Db2 utilities support the use of extended address volumes (EAV) for
VSAM data sets and extended format (EF) sequential data sets.

Restriction: Db2 does not support the undefined record format (RECFM=U) for
any data set.

Related information:
- Data set record formats (z/OS basic skills)
- DEVSUPxx (device support options) (MVS Initialization and Tuning Reference)
- Large Block Interface (LBI) (z/OS DFSMS Using Data Sets)
- DCB Subparameters (MVS JCL Reference)
- Extended Address Volumes (z/OS DFSMS Using Data Sets)

Sort work data sets

 Allocate sort work data sets dynamically for automatic adjustment to the amount
 of data being processed by the utility. To facilitate dynamic allocation of sort work,
specify a valid disk device type for the SORTDEVT parameter. Tape devices are not
 supported by the sort program.

Data set concatenation

When you use Db2 utilities, you can concatenate input data sets. The data sets in a
concatenation list can have different block sizes, logical record lengths, and record
formats. If you want to concatenate variable and fixed-blocked data sets, the logical
record length must be 8 bytes smaller than the block size.

You cannot concatenate output data sets.

Related information:
Concatenating Data Sets (MVS JCL User’s Guide)

Data set disposition

Because you might need to restart a utility, define your data sets as follows:

- Use DISP=(NEW,CATLG,CATLG) or DISP=(MOD,CATLG) for data sets that you want to retain.
- Use DISP=(MOD,DELETE,CATLG) for data sets that you want to discard after utility execution.
- Use DISP=(NEW,DELETE) for the SORTWKnn data sets for your sort program, or refer to the documentation for your sort program for alternatives.
- Do not use temporary data set names.

Related information:
- DISP Parameter (MVS JCL Reference)
- DFSORT Application Programming Guide
- Db2 Sort for z/OS

Preventing unauthorized access to data sets

To prevent unauthorized access to data sets (for example, image copies), you can protect the data sets with the Resource Access Control Facility (RACF) licensed program. To use a utility with a data set that is protected by RACF, you must be authorized to access the data set.

Related information:
- Protecting data sets through RACF (Managing Security)

Utility control statements

Utility control statements define the function that the utility job performs.

For example, the following LOAD utility control statement specifies how tables are to be loaded into the DSN8B10.DEPT table:

```
LOAD
  INTO TABLE DSN8B10.DEPT PART 1 REPLACE
```

You can create a utility control statement with the ISPF/PDF edit function and save it in a sequential or partitioned data set.

Db2 typically reads utility control statements from the SYSIN in-stream data set. Db2 can read LISTDEF control statements from the SYSLISTD data set and TEMPLATE control statements from the SYSTEMPL data set.

Control statement coding rules

Utility control statements must obey the following rules:

- **Fixed-length records**: If the records are 80-character fixed-length records, Db2 ignores columns 73 through 80.
- **Statements that span records**: The records are concatenated before they are parsed; therefore, a statement or any of its syntactical constructs can span more than one record. No continuation character is necessary.
However, if the input data set contains variable-length records, Db2 might interpret the part of a statement that is in column 1 as the continuation of the statement from the previous record. To avoid syntax errors, ensure that all syntactical constructs in utility control statements are properly delimited. Doing so is especially important for the first character in each record of a data set with variable-length records.

- **Character set encoding:** All utility control statements in a data set must be written entirely in a single character set. They can be written in EBCDIC (code page 500) or Unicode UTF-8 (code page 1208). Db2 automatically detects and processes Unicode UTF-8 control statements if the first character of the data set is one of the following characters:
  - A Unicode UTF-8 blank (x'20')
  - A Unicode UTF-8 dash (x'2D')
  - A Unicode UTF-8 uppercase A through Z (x'41' through x'5A')

In all other cases, the control statement data set is processed as EBCDIC. Db2 issues an informational message (DSNU1044I or DSNU1045I) to identify the character set that it is using to process the statement.

Avoid the following EBCDIC characters as the first character in the input data set:

- `. < ( + | &`

These EBCDIC characters have the same hexadecimal code point value as the Unicode UTF-8 characters ] to F. Therefore, the control statement can be misinterpreted as Unicode. This misinterpretation causes a utility syntax error. However, these characters might cause Unicode to EBCDIC translation errors and abends before the syntax error is detected. In this case, after the syntax error is detected, message DSNU005I might contain indecipherable statements and message DSNU082I might identify an indecipherable keyword.

**Related information:**

- [Code pages and CCSIDs (Db2 Internationalization Guide)](http://example.com/codepages)
- [EBCDIC (Db2 Internationalization Guide)](http://example.com/ebcdic)
- [Unicode (Db2 Internationalization Guide)](http://example.com/unicode)
- [DSNU1044I (Db2 Messages)](http://example.com/dsn1044i)
- [DSNU1045I (Db2 Messages)](http://example.com/dsn1045i)
- [DSNU005I (Db2 Messages)](http://example.com/dsn005i)
- [DSNU082I (Db2 Messages)](http://example.com/dsn082i)

- **Syntax:** The utility statement must start with the syntax for a utility. Other syntactical constructs in the utility control statement describe options. You can separate these constructs with an arbitrary number of blanks.

- **Options:** The options that you can specify after the utility name depend on which online utility you use. To specify a utility option, specify the option keyword, followed by its associated parameter or parameters, if any. The parameter value can be a keyword. You need to enclose the values of some parameters in parentheses. The syntax diagrams for utility control statements show parentheses where they are required.

Where the syntax of each utility control statement is described, parameters are indented under the option keyword that they must follow. The following option is a typical example:

**WORKDDN** *ddname*

Specifies a temporary work file.

*ddname* is the data set name of the temporary file.
The default value is **SYSUT1**.

In the example, WORKDDN is an option keyword, and *ddname* is a variable parameter. You can enclose parameter values in parentheses, but parentheses are not always required. In this case, you can specify the temporary work file as either WORKDDN SYSUT1 or WORKDDN (SYSUT1).

- **Delimiter character and decimal character:** In a utility control statement, when you specify multiple numeric values that are meant to be delimited, you must delimit these values with a comma (","). You must use this delimiter regardless of the definition of DECIMAL in the application defaults load module (either DSNHDECP or a user-specified application defaults load module). Likewise, when you specify a decimal number in a utility control statement, you must use a period ("."), regardless of the definition of DECIMAL in the application defaults load module.

**Related information:**

- [DECIMAL POINT IS field (DECIMAL DECP value) (Db2 Installation and Migration)]

- **Multiple utility control statements:** You can specify more than one utility control statement in the SYSIN stream. However, if any of the control statements return a return code of 8 or greater, the subsequent statements in the job step are not executed.

- **Comments:** You can enter comments within the SYSIN stream. Comments must begin with two hyphens ("--") and are subject to the following rules:
  - You must use two hyphens on the same line with no space between them.
  - You can start comments wherever a space is valid, except within a delimiter token.
  - The end of a line terminates a comment.

For example, the following statement shows two comments:

```plaintext
// SYSS IN DD *
RUNSTATS TABLESPACE DSNDB06.SYSDDF -- COMMENT HERE
-- COMMENT HERE
/*
```

- **The concatenation operator:** Utility control statements support the `||` concatenation operator. The operator is allowed between two delimited character strings or between two non-delimited character strings. (Delimited character strings are enclosed in double quotation marks.) The result is a character string that consists of the string that is after the operator concatenated to the string that precedes the operator.

For example, the following statement shows the concatenation operator:

```
string1 || string2
```

Both *string1* and *string2* must be syntactically correct within each SYSIN input record. Quotation marks must be balanced within each string. If DBCS characters are used, shift-out and shift-in characters must be balanced within each string. Any one multi-byte character must be contained entirely within a single SYSIN record.

The `||` operator must be entered as a stand-alone token, with one or more blanks before and after it. It can be entered on the same input record as "*string1*", alone on an input record, or on the same input record with "*string2*". This operator functions at the token level before any context is detected or semantic meaning is applied. An example utility statement is shown in the following statement:
The || operator in an EXEC SQL control statement is ignored by utility processing, because the operator has an existing SQL meaning. The operators remain part of the SQL statement for subsequent processing by SQL. Another example of the || concatenation operator is shown in the following statement:

```sql
LOAD INTO TABLE
CREA ||
TOR.
"TABL" ||
"ENAME"
```

In this example, the strings CREA and TOR are non-delimited, and the strings TABL and ENAME are delimited by double quotation marks. The processed output of this example is equivalent to the following statement:

```sql
LOAD INTO TABLE CREATOR."TABLENAME"
```

**Tips for using multi-byte character sets in control statements**

Multi-byte character sets can be difficult to work with in fixed 80-byte SYSIN data sets. Long object names and long character literals might not fit on a single line.

Where possible, avoid having to break object names or character literals by using the following techniques:

- Use a SYSIN data set with variable length records or sufficiently large record length.
- Use shorter object names. The longer the name, the more likely continuation issues arise.
- If possible, process the object by space name (table space or index space) and avoid specifying long multi-byte table and index names in utility syntax.

If necessary, use one of the following continuation techniques:

- Shift the starting point of the string to the left or right within the input record so that a complete multi-byte character ends in column 72. Continue with the next character in column 1 of the next input record.
- Separate qualified object names into two parts following the dot ".", which separates the qualifiers. Separating long names into multiple parts makes it easier to follow the continuation rules. This technique cannot be used in the EXEC SQL utility, which must follow both utility and SQL syntax rules.
- Use the || concatenation operator to divide long identifiers into two or more parts that fit into each SYSIN record.

---

**Invoking an online utility by using the DB2 Utilities panel in DB2I**

One way to invoke Db2 online utilities is to use the DB2 Utilities panel. Consider using this method if you do not have much JCL knowledge.

To use this method, you must have TSO and access to the Db2 Utilities Panel in Db2 Interactive (DB2I).

**Restriction:** You cannot use the DB2 Utilities panel in DB2I to submit jobs for the following utilities:

- BACKUP SYSTEM
• COPYTOCOPY
• RESTORE SYSTEM
• COPY for a list of objects if templates are not used for the output data sets

Procedure

To invoke an online utility by using the DB2 Utilities panel in DB2I:

1. Create the utility control statement for the online utility that you want to invoke, and save it in a sequential or partitioned data set.

2. From the ISPF Primary Option Menu, select DB2 until the DB2I Primary Option Menu is displayed. (Depending on how your menus are set up, you might need to select DB2 from several menus until the DB2I Primary Option Menu is displayed.)

3. On the DB2I Primary Option Menu, select the UTILITIES option.

4. On the DB2 Utilities panel, complete the following fields and press Enter::

   1 FUNCTION
   EDITJCL

   3 UTILITY
   The name of the utility that you want to run

   4 STATEMENT DATA SET
   The name of the data set that contains the utility control statement.

   For more information about the fields on this panel, see “DB2 Utilities panel” on page 23

5. If the Data Set Names panel is displayed, complete the relevant fields with the data set names and press Enter. This panel is displayed if you specified COPY, LOAD, MERGECOPY, REORG TABLESPACE, or UNLOAD. For more information about the fields on this panel, see “Data Set Names panel” on page 24

6. If the Control Statement Data Set Names panel is displayed, complete the fields on that panel and press Enter. This panel is displayed if you specified that you want to use templates or lists on the DB2 Utilities panel. For more information about the fields on this panel, see “Control Statement Data Set Names panel” on page 25

7. Edit the generated JCL as needed and type SUBMIT or SUB on the editor command line.

Results

You can view the output of the job as you normally view the output of any batch job on your system. One common way is by using System Display and Search Facility (SDSF).

Related concepts:

“The Utility control statements” on page 18

What is SDSF? (z/OS basic skills)

Related reference:

Chapter 18, “LISTDEF,” on page 215
Chapter 34, “TEMPLATE,” on page 793
DB2 Utilities panel

Use the DB2 Utilities panel to invoke utilities without writing JCL.

Fields that you must specify on this panel are highlighted in the following figure:

```
DSNEUP01  DB2 UTILITIES
===>
Select from the following:

1 FUNCTION ===> EDITJCL (SUBMIT job, EDITJCL, DISPLAY, TERMINATE)
2 JOB ID ===> TEMP (A unique job identifier string)
3 UTILITY ===> COPY (CHECK DATA, CHECK INDEX, CHECK LOB,
                        COPY, DIAGNOSE, LOAD, MERGE, MODIFY,
                        QUIESCE, REBUILD, RECOVER, REORG INDEX,
                        REORG LOB, REORG TABLESPACE, REPORT,
                        REPAIR, RUNSTATS, STOSPACE, UNLOAD.)
4 STATEMENT DATA SET ===> UTIL

Specify restart or preview option, otherwise specify NO.

5 RESTART ===> NO (NO, CURRENT, PHASE or PREVIEW)
6 LISTDEF? (YES|NO) ===> NO TEMPLATE? (YES|NO) ===> NO
7 LIB ===> (BLANK or DB2 LIB NAME)

* The data set names panel will be displayed when required by a utility.

PRESS: ENTER to process END to exit HELP for more information
```

Figure 1. Db2 Utilities panel

1 FUNCTION

Specify one of the following options:

**SUBMIT**

The generated JCL is submitted for processing after you complete this panel and any subsequent panels. You cannot edit the JCL.

**EDITJCL**

You can edit the generated JCL before you submit the utility job. For example, you might want to add or edit parameters on the JOB statement or add a ROUTE statement.

The other two options for this field, DISPLAY and TERMINATE, are not part of invoking a utility. DISPLAY performs the same function as the DISPLAY UTILITY command; it shows the status of a utility job. TERMINATE performs the same function as the TERM UTILITY command; it terminates a utility.

2 JOB ID

Specify a unique identifier for your utility job. The default value is TEMP.

3 UTILITY

Specify the utility that you want to run.

4 STATEMENT DATA SET

Specify the data set that contains the utility control statement. TSO adds your user ID as a prefix unless you enclose the data set name between apostrophes. The default data set is UTIL.
5 RESTART
If you want to restart a stopped utility, specify CURRENT or PHASE. If you want to run a utility in PREVIEW mode, specify PREVIEW. Otherwise, leave the default value of NO.

6 LISTDEF? TEMPLATE?
Specify whether you are using LISTDEF statements or TEMPLATE statements for this utility job.

7 LIB
If you want the generated JCL to use a Db2 subsystem library other than the default, specify the data set name of the Db2 subsystem library.

Data Set Names panel
Use the Data Set Names panel to provide the relevant data set names when invoking utilities by using DB2I.

This panel is displayed if you specified COPY, LOAD, MERGECOPY, REORG TABLESPACE, or UNLOAD on the DB2 Utilities panel as the utility that you want to invoke.

<table>
<thead>
<tr>
<th>DSNEUPO2</th>
<th>DATA SET NAMES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>====&gt;</td>
</tr>
<tr>
<td></td>
<td>Enter data set name for LOAD or REORG TABLESPACE:</td>
</tr>
<tr>
<td>1 RECDSN</td>
<td>===&gt;</td>
</tr>
<tr>
<td></td>
<td>Enter data set name for LOAD, REORG TABLESPACE or UNLOAD:</td>
</tr>
<tr>
<td>2 DISCDSN</td>
<td>===&gt;</td>
</tr>
<tr>
<td></td>
<td>Enter output data sets for local/current site for COPY, MERGECOPY, LOAD, or REORG:</td>
</tr>
<tr>
<td>3 COPYDSN</td>
<td>===&gt;</td>
</tr>
<tr>
<td>4 COPYDSN2</td>
<td>===&gt;</td>
</tr>
<tr>
<td></td>
<td>Enter output data sets for recovery site for COPY, LOAD, or REORG:</td>
</tr>
<tr>
<td>5 RCPYDSN1</td>
<td>===&gt;</td>
</tr>
<tr>
<td>6 RCPYDSN2</td>
<td>===&gt;</td>
</tr>
<tr>
<td>7 PUNCHDSN</td>
<td>===&gt;</td>
</tr>
<tr>
<td></td>
<td>PRESS: ENTER to process END to exit HELP for more information</td>
</tr>
</tbody>
</table>

Figure 2. Data Set Names panel

1 RECDSN
Complete this field if you are running LOAD, REORG TABLESPACE, or UNLOAD. For LOAD, specify the name of the data set that contains the records that are to be loaded. For REORG TABLESPACE or UNLOAD, specify the unload data set.

2 DISCDSN
If you are running LOAD or REORG TABLESPACE with discard processing, specify a discard data set.

3 COPYDSN
If you are running COPY, LOAD, or REORG TABLESPACE, specify the name of the primary output data set for the local site. If you are running MERGECOPY, specify the name of the primary output data set for the current site. The DD name that the panel generates for this field is SYSCOPY. This field is optional for the following jobs:
• LOAD
• REORG TABLESPACE with the SHRLEVEL NONE option

This field is required for the following jobs:
• COPY
• MERGECOPY
• REORG TABLESPACE with the SHRLEVEL REFERENCE or SHRLEVEL CHANGE options.

4 COPYDSN2
If you are running COPY, LOAD, or REORG TABLESPACE, specify the name of the backup output data set for the local site. If you are running MERGECOPY, specify the name of the backup output data set for the current site. The DD name that the panel generates for this field is SYSCOPY2.

5 RCPYDSN1
If you are running COPY, LOAD, or REORG TABLESPACE, specify the name of the primary output data set for the recovery site. The DD name that the panel generates for this field is SYSRCOPY1.

6 RCPYDSN2
If you are running COPY, LOAD, or REORG TABLESPACE, specify the name of the backup output data set for the recovery site. The DD name that the panel generates for this field is SYSRCOPY2.

7 PUNCHDSN
If you are running any of the following utility jobs, specify the name of the output data set for the generated LOAD utility control statements:
• REORG TABLESPACE with the UNLOAD EXTERNAL or DISCARD options
• UNLOAD

The DD name that the panel generates for this field is SYSPUNCH.

Related reference:

“DB2 Utilities panel” on page 23

Control Statement Data Set Names panel

Use the Control Statement Data Set Names panel to provide the data set names for LISTDEF and TEMPLATE statements when invoking utilities by using DB2I.

This panel is displayed if you specified YES in the LISTDEF field or TEMPLATE field on the DB2 Utilities panel. All of the fields on this panel are optional.

Figure 3. Control Statement Data Set Names panel
1 LISTDEF DSN
Specify the name of the data set that contains a LISTDEF control statement. The default is the SYSIN data set. This field is ignored if you specified NO in the LISTDEF field in the DB2 Utilities panel.

2 TEMPLATE DSN
Specify the name of the data set that contains a TEMPLATE control statement. The default is the SYSIN data set. This field is ignored if you specified NO in the TEMPLATE field in the DB2 Utilities panel.

Related reference:
“DB2 Utilities panel” on page 23

Invoking a Db2 utility by using the DSNU CLIST command in TSO

One way to run a Db2 online utility is to invoke the DSNU CLIST command under TSO. This command generates a data set with JCL that invokes the DSNUPROC procedure to execute utilities as batch jobs.

Restriction: You cannot use the DSNU CLIST command for a COPY job with a list of objects.

Before you begin

Ensure that the Db2 CLIST library (prefix.SDSNCLST) is included in the SYSPROC DD statement in the startup procedure for your TSO session.

About this task

The DSNU CLIST command generates a job that performs only one utility operation. If you need to run more than one utility in a single job, you can invoke the DSNU CLIST command for each utility that you need; then edit and merge all of the output into one job or step.

Procedure

To invoke a Db2 utility by using the DSNU CLIST command in TSO:

1. Create a data set or member that contains the utility control statement.

2. Execute the DSNU CLIST command. See “DSNU CLIST command syntax” on page 27. If you want to be able to edit the generated JCL, specify EDIT(SPF) or EDIT(TSO) on the command.
   DSNU generates the JCL in a data set that is named DSNUxxx.CNTL, where DSNUxxx is the control file name for the utility. See Table 1 on page 31. This output data set contains the statements that are necessary to invoke the DSNUPROC procedure, which, in turn, executes the utility. See "DSNU CLIST command output" on page 32.

3. Optional: Edit the generated JCL data set.
   You can add a JCL statement to the job stream, change JCL parameters (such as DD names), or change utility control statements.
   If your utility control statement specifies a DD name value that is not the default value, you must change the DD name in the generated JCL. For example, in the REORG TABLESPACE utility statement, the UNLDDN option specifies the name of the unload data set. The default value for UNLDDN is the DD name SYSREC. Therefore, DSNU CLIST builds a SYSREC DD statement
for REORG TABLESPACE. If you specify a different value for UNLDDN in the utility statement, you must edit the JCL data set and change SYSREC to the DD name value that you used.

4. Optional: Rename the JCL data set.

DSNU CLIST reuses the DSNUxxx.CNTL data set for any subsequent jobs with the same utility name. If you want to submit more than one job that executes the same utility, rename the JCL data sets and submit them separately.

**Examples**

**Example 1: Invoking the DSNU CLIST command for the REORG TABLESPACE utility**

The following DSNU CLIST command generates a data set, authorization-id.DSNURGT.CNTL, that contains JCL statements. That JCL invokes the DSNUPROC procedure, which in this case invokes the REORG TABLESPACE utility.

```bash
%DSNUUTILITY(REORG TABLESPACE) INDSN(MYREOR.DATA)RECDN(MYREOR.WORK) RESTART(NO)EDIT(TSO)SUBMIT(YES)
```

The MYREOR.DATA data set contains the REORG TABLESPACE utility statement. MYREOR.WORK is a temporary data set that is required by REORG TABLESPACE. The TSO editor is invoked to allow editing of the JCL data set, authorization-id.DSNURGT.CNTL. After the data set is edited, the TSO editor then submits the JCL data set as a batch job. This JCL data set is not modified by this CLIST command statement until a new request is made to execute the REORG TABLESPACE utility.

**Example 2: Invoking the DSNU CLIST command for the COPY utility**

The following example shows how to invoke the DSNU CLIST command for the COPY utility.

```bash
%DSNU
 UTILITY (COPY)
 INDSN ('MYCOPY(STATEMNT)')
 COPYDSN ('MYCOPIES.DSN8D11A.JAN1')
 EDIT (TSO)
 SUBMIT (YES)
 UID (TEMP)
 RESTART (NO)
```

**Related reference:**

"The supplied JCL procedure (DSNUPROC) for invoking a Db2 online utility” on page 33

**DSNU CLIST command syntax**

You can execute the DSNU CLIST command from the TSO command processor or from the DB2I Utilities panel.
DSNU CLIST option descriptions

The parentheses that are shown in the following descriptions are required.

% Identifies DSNU as a member of a command procedure library. Specifying this parameter is optional; however, it does improve performance.

Notes:
1. All data sets must be cataloged.
UTILITY (*utility-name*)
Specifies the utility that you want to execute.

For a list of the online utilities that you can specify, see Table 1 on page 31.

INDSN(*data-set-name*(member-name*))
Specifies the name of the data set that contains the utility statements. If the data set is partitioned, also specify the member name (member-name).

Do not specify a data set or member that includes double-byte character set (DBCS) data.

CONTROL(*control-option:* ...)
Specifies whether to trace the CLIST command execution.

The default value is NONE, which omits tracing. NONE generates a CONTROL statement with the options NOLIST, NOCONLIST, and NOSYMLIST.

Otherwise, you can specify one or more of the following options. Separate items in the list by colons (:). To abbreviate, specify only the first letter of the option.

**LIST** Displays TSO commands after symbolic substitution and before command execution.

**CONLIST** Displays CLIST commands after symbolic substitution and before command execution.

**SYMLIST** Displays all executable statements (TSO commands and CLIST statements) before the scan for symbolic substitution.

**DB2I**
Indicates whether the DSNU CLIST command is being called from the DB2I environment.

(NO) The command is not being called from the DB2I environment.

(YES) The command is called from the DB2I environment.

Only the DB2I utility panels are to execute the CLIST command with DB2I(YES).

DISCDSN (*data-set-name*)
The name of the data set to be used as a discard data set. DISCDSN applies only to the LOAD and REORG utilities. For LOAD, this data set holds records that are not loaded. For REORG, this data set holds records that are not reloaded.

COPYDSN(*data-set-name*)
The name of the data set to be used as the output data set. COPYDSN is required for COPY, MERGECOPY, and REORG with SHRLEVEL REFERENCE or CHANGE. COPYDSN is optional for LOAD and REORG with SHRLEVEL NONE.

COPYDSN2 (*data-set-name*)
The name of the data set to be used for the backup copy. COPYDSN2 is optional for COPY, MERGECOPY, LOAD, and REORG.
**RCPYDSN1** *(data-set-name)*
The name of the data set to be used for the remote-site primary copy. RCPYDSN1 is optional for COPY, LOAD, and REORG.

**RCPYDSN2** *(data-set-name)*
The name of the data set to be used for the remote-site backup copy. RCPYDSN2 is optional for COPY, LOAD, and REORG.

**RECDSN** *(data-set-name)*
The name of the data set that LOAD is to use for input or that REORG TABLESPACE or UNLOAD is to use as the unload data set. RECDSN is required for LOAD and REORG TABLESPACE.

**PUNCHDSN** *(data-set-name)*
The name of the data set that REORG or UNLOAD is to use to hold the generated LOAD utility control statements for UNLOAD EXTERNAL or DISCARD.

**EDIT**
Specifies whether to invoke an editor so that you can edit the temporary file that the CLIST command generates.

*NO*  
Does not invoke an editor.

*SPF*  
Invokes the ISPF editor.

*TSO*  
Invokes the TSO editor.

**RESTART**
Specifies whether this job restarts a current utility job, and, if so, at what point it is to be restarted.

*NO*  
Indicates that the utility is a new job, not a restarted job. The utility identifier (UID) must be unique for each utility job step.

*CURRENT*  
Restarts the utility at the most recent commit point.

*PHASE*  
Restarts the utility at the beginning of the current stopped phase. You can determine the current stopped phase by issuing the DISPLAY UTILITY command.

*PREVIEW*  
Restarts the utility in PREVIEW mode. While in PREVIEW mode, the utility checks for syntax errors in all utility control statements, but normal utility execution does not take place.

**SUBMIT**
Specifies whether to submit the generated JCL for processing.

*NO*  
Does not submit the JCL data set for processing.

*YES*  
Submits the JCL data set for background processing by using the TSO SUBMIT command.

*PROMPT*  
Prompts you, after the data set is processed, to specify whether to submit
the JCL data set for batch processing. You cannot use PROMPT when the CLIST command is executed in the TSO batch environment.

**SYSTEM (subsystem-name)**
Specifies the Db2 subsystem or group attachment name or subgroup attachment name. The default value is **DSN**.

**UID (utility-id)**
Provides a unique identifier for this utility job within Db2.

Do not reuse the utility ID of a stopped utility that has not yet been terminated, unless you want to restart that utility. If you do use the same utility ID to invoke a different utility, Db2 tries to restart the original stopped utility with the information that is stored in the SYSUTIL directory table.

The default value is `tso-userid.control-file-name`, where `tso-userid` is your TSO user ID and `control-file-name` is one of the following values:

<table>
<thead>
<tr>
<th>Utility</th>
<th>control-file-name</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHECK INDEX</td>
<td>DSNUCHI</td>
</tr>
<tr>
<td>CHECK DATA</td>
<td>DSNUCHD</td>
</tr>
<tr>
<td>CHECK LOB</td>
<td>DSNUCHL</td>
</tr>
<tr>
<td>COPY</td>
<td>DSNUCOP</td>
</tr>
<tr>
<td>DIAGNOSE</td>
<td>DSNUDIA</td>
</tr>
<tr>
<td>LOAD</td>
<td>DSNULOA</td>
</tr>
<tr>
<td>MERGECOPY</td>
<td>DSNUMER</td>
</tr>
<tr>
<td>MODIFY</td>
<td>DSNUMOD</td>
</tr>
<tr>
<td>QUIESCE</td>
<td>DSNUQUI</td>
</tr>
<tr>
<td>REBUILD INDEX</td>
<td>DSNUREB</td>
</tr>
<tr>
<td>RECOVER</td>
<td>DSNUREC</td>
</tr>
<tr>
<td>REORG INDEX</td>
<td>DSNURGI</td>
</tr>
<tr>
<td>REORG LOB</td>
<td>DSNURGL</td>
</tr>
<tr>
<td>REORG TABLESPACE</td>
<td>DSNURGT</td>
</tr>
<tr>
<td>REPAIR</td>
<td>DSNUREP</td>
</tr>
<tr>
<td>REPORT</td>
<td>DSNURPT</td>
</tr>
<tr>
<td>RUNSTATS</td>
<td>DSNURUN</td>
</tr>
<tr>
<td>STOSPACE</td>
<td>DSNUSTO</td>
</tr>
<tr>
<td>UNLOAD</td>
<td>DSNUUNL</td>
</tr>
</tbody>
</table>

**UNIT (unit-name)**
Assigns a unit address, a generic device type, or a user-assigned group name for a device on which a new temporary or permanent data set resides. When the CLIST command generates the JCL, it places `unit-name` after the UNIT clause of the generated DD statement.

The default value is **SYSDA**.

**VOLUME (vol-ser)**
Assigns the serial number of the volume on which a new temporary or permanent data set resides. When the CLIST command generates the JCL, it
places vol-ser after the VOL=SER clause of the generated DD statement. If you omit VOLUME, the VOL=SER clause is omitted from the generated DD statement.

**LIB (data-set-name)**

Specifies the data set name of the Db2 subsystem library. The value that you specify is used as the LIB parameter value when the DSNUPROC JCL procedure is invoked.

**Related reference:**

“DSNU CLIST command output”

---

**DSNU CLIST command output**

The DSNU CLIST command builds a one-step job stream. The generated JCL consists of a JOB statement, an EXEC statement that executes the Db2 utility processor, and the required DD statements. The JOB statement also includes the SYSIN DD * job stream. You can edit any of these statements.

The following figure shows example JCL that the DSNU CLIST command generates before any editing:

```plaintext
//DSNUCOP JOB your-job-statement-parameters
// USER=userid, PASSWORD=password
//*ROUTE PRINT routing-information
//*UTIL EXEC DSNUPROC, SYSTEM=DSN, UID=TEMP, UTPROC=''
//SYSCOPY DD DSN=MYCOPIES.DSN8D11A.JAN1, DISP=(MOD, CATLG, CATLG),
// UNIT=SYSDA, SPACE=(CYL, (1,1))
//SYSIN DD *
COPY TABLESPACE DSN8D11A.DSN8S11D
   FULL NO
   SHRLEVEL CHANGE
/*
Figure 4. Control file DSNUCOP.CNTL

The JCL includes the following statements:

**JOB**

The CLIST command uses any JOB statements that you saved when using DB2I. If no JOB statements exist, Db2 produces a skeleton JOB statement that you can modify. The job name is DSNU, followed by the first three letters of the name of the utility that you are using.

**EXEC**

The CLIST command builds the EXEC statement to invoke DSNUPROC. The values that you specified for the SYSTEM, UID, and RESTART options for the DSNU CLIST command become the values of the SYSTEM, UID, and UTPROC parameters for DSNUPROC.

**DD**

The CLIST command also builds the necessary JCL DD statements. The following DD statements are always generated by the CLIST command:

**SYSPRINT DD SYSOUT=A**

Defines OUTPUT, SYSPRINT as SYSOUT=A. Utility messages are sent to the SYSPRINT data set. You can use the TSO ALLOCATE command to control the disposition of the SYSPRINT data set. For example, you can send the data set to your terminal.

**UTPRINT DD SYSOUT=A**

Defines UTPRINT as SYSOUT=A. If any utility requires a sort, it executes the sort program. Messages from that program are sent to UTPRINT.
SYSIN DD *

Defines SYSIN. To build the SYSIN DD * job stream, DSNU copies the
data set that is named by the INDSN parameter. The INDSN data set
does not change, and you can reuse it when the DSNU procedure
finishes running.

Other DD statements are generated depending on the utility that you are
executing.

Related reference:

- Db2 Sort for z/OS
- ALLOCATE command (TSO/E Command Reference)

Related information:

- DFSORT Application Programming Guide

The supplied JCL procedure (DSNUPROC) for invoking a Db2 online utility

One way to invoke a Db2 online utility is to execute the supplied JCL procedure
DSNUPROC. The advantage of using DSNUPROC is that it limits the amount of
JCL that you need to write.

DSNUPROC is a standard set of JCL that IBM provides to invoke online utilities.
You customize this JCL procedure at installation time (in SYS1.PROCLIB).

Executing DSNUPROC

To execute the DSNUPROC procedure, write and submit a JCL job that contains
the following statements:

- An EXEC statement that specifies DSNUPROC and any parameters that you
  want. Possible parameters are listed in “DSNUPROC syntax” on page 34.
- A SYSIN DD statement for the utility control statements.
- DD statements for any other data sets that your utility job needs except for
  SYSPRINT and UTPRINT. DSNUPROC provides the SYSPRINT and UTPRINT
  DD statements for printed output.

DSNUPROC uses the parameters that you supply to build an EXEC statement that
executes an online utility.
DSNUPROC syntax

```
DSNUPROC
   LIB=prefix.SSPGM
   ,LIB=DB2library-name
   ,SIZE=OM
   ,SIZE=region-size
   ,SYSTEM=DSN
   ,SYSTEM=subsystem-name
   ,UID=utility-qualifier
   ,UTPROC='RESTART'
   ,UTPROC='RESTART(CURRENT)'
   ,UTPROC='RESTART(PHASE)'
   ,UTPROC='PREVIEW'
```

DSNUPROC option descriptions

The following list describes all the parameters.

**LIB=**
Specifies the data set name of the Db2 subsystem library.

The default value is `prefix.SSPGM`.

**SIZE=**
Specifies the region size of the utility execution area; that is, the value represents the number of bytes of virtual storage that are allocated to this utility job.

The default value is `0M`.

**SYSTEM=**
Specifies the Db2 subsystem or group attachment name or subgroup attachment name.

The default value is `DSN`.

**UID=**
Specifies the unique identifier for your utility job. The maximum name length is 16 characters. If the name contains special characters, enclose the entire name between apostrophes (for example, 'PETERS.JOB').

Do not reuse the utility ID of a stopped utility that has not yet been terminated. If you do use the same utility ID to invoke a different utility, Db2 tries to restart the original stopped utility with the information that is stored in the SYSUTIL directory table.

The default is an empty string. If you do not specify UID, or you specify UID='', Db2 assigns a utility ID of the form `user-id.job-name` to the utility job.

The total length of the generated utility ID is longer than 16 bytes, the user ID is truncated to 7 bytes.

**UTPROC=**
Controls restart processing. The default is an empty string. Use the default if you want to allow Db2 to perform default restart processing.

To override the default restart behavior, specify one of the following values:

`'RESTART'`
Restarts at the most recent commit point.
'RESTART(CURRENT)'  
Restarts at the most recent commit point. RESTART and RESTART(CURRENT) have the same meaning.

'RESTART(PHASE)'  
Restarts at the beginning of the phase that was being processed.

'PREVIEW'  
Restarts in preview mode. While in preview mode, the utility checks for syntax errors in all utility control statements, but normal utility execution does not take place.

Related information:  
“Restarting an online utility” on page 44

Sample DSNUPROC listing

You can use JCL similar to the following example to invoke DSNUPROC:

```jcl
//DSNUCOP JOB your-job-statement-parameters  
// USER=userid, PASSWORD=password  
//*ROUTE PRINT routing-information  
//UTIL EXEC DSNUPROC, SYSTEM=DSN, UID=TEMP, UTPROC='  
//SYSCOPY DD DSN=MYPYCOPIES.DSN8D11A.JAN1, DISP=(MOD, CATLG, CATLG),  
// UNIT=SYSDA, SPACE=(CYL, (1, 1))  
//SYSIN DD *  
COPY TABLESPACE DSN8D11A.DSN8S11D  
FULL NO  
SHRLEVEL CHANGE
/*
```

The preceding JCL job (DSNUCOP) executes the following DSNUPROC procedure:

```jcl
//DSNUPROC PROC LIB='DSN10.SDSNLOAD',  
// SYSTEM=DSN,  
// SIZE=OK, UID='', UTPROC=''  
/* PROCEDURE-NAME: DSNUPROC *  
/* DESCRIPTIVE-NAME: UTILITY PROCEDURE *  
/* FUNCTION: THIS PROCEDURE INVOKES THE ADMF UTILITIES IN THE *  
/* BATCH ENVIRONMENT *  
/* PROCEDURE-OWNER: UTILITY COMPONENT *  
/* COMPONENT-INVOKED: ADMF UTILITIES (ENTRY POINT DSNUTILB). *  
/* ENVIRONMENT: BATCH *  
/* INPUT: *  
/* PARAMETERS: *  
/* LIB = THE DATA SET NAME OF THE DB2 PROGRAM LIBRARY. *  
/* THE DEFAULT LIBRARY NAME IS PREFIX.SDSNLOAD, *  
/* WITH PREFIX SET DURING INSTALLATION. *  
/* SIZE = THE REGION SIZE OF THE UTILITIES EXECUTION AREA. *  
/* THE DEFAULT REGION SIZE IS 2048K. *  
/* SYSTEM = THE SUBSYSTEM NAME USED TO IDENTIFY THIS JOB TO DB2. THE DEFAULT IS "DSN". *  
/* UID = THE IDENTIFIER WHICH WILL DEFINE THIS UTILITY JOB TO DB2. IF THE PARAMETER IS DEFAULTED OR *  
/* SET TO A NULL STRING, THE UTILITY FUNCTION WILL *  
/* USE ITS DEFAULT, USERID.JOBNAME. EACH UTILITY *  
/* WHICH HAS STARTED AND IS NOT YET TERMINATED *
```

Chapter 6. Invoking Db2 online utilities  
35
Invoking a Db2 online utility by creating the JCL data set yourself

Db2 online utilities run as standard z/OS jobs. You can either write the JCL for these jobs yourself or use tools to generate some or all of the JCL for you.

About this task

To limit the amount of JCL that you need write, you can use “The supplied JCL procedure (DSNUPROC) for invoking a Db2 online utility” on page 33.

Procedure

To invoke a Db2 online utility by creating the JCL data set yourself:

Include the following JCL statements:

- The JOB statement that is required by your installation standards.
- The JOBLIB or STEPLIB DD statements that are required to access Db2.
- An EXEC statement.

The statement can specify a procedure that contains the required JCL, such as DSNUPROC, or it can be of the following form:

```
//stepname  EXEC  PGM=DSNUTILB,PARM='system,[uid],[utproc]'  
```

The brackets, [ ], indicate optional parameters. The parameters have the following meanings:
DSNUTILB
The utility control program. The program must reside in an APF-authorized library.

**system**
The Db2 subsystem.

**uid**
The unique identifier for your utility job.

Do not reuse the utility ID of a stopped utility that has not yet been terminated. If you do use the same utility ID to invoke a different utility, Db2 tries to restart the original stopped utility with the information that is stored in the SYSIBM.SYSUTIL directory table.

**utproc**
The restart processing behavior. Specify this option only when you want to restart the utility job. You can specify one of the following values:

'**RESTART**'
Restarts the utility at the most recent commit point. This option has the same meaning as 'RESTART(CURRENT).'</n

'**RESTART(CURRENT)**'
Restarts the utility at the most recent commit point. This option has the same meaning as 'RESTART.'

'**RESTART(PHASE)**'
Restarts the utility at the beginning of the phase that executed most recently.

'**RESTART(PREVIEW)**'
Restarts the utility in preview mode. While in preview mode, the utility checks for syntax errors in all utility control statements, but normal utility execution does not take place.

**Example EXEC statement:**

```
//stepname
EXEC PGM=DSNUTILB,PARM='DSN,TEMP'
```

- Any required DD statements. For information about the DD statements that each utility needs, see “Data sets that online utilities use” on page 16.

**Related concepts:**
- Basic JCL concepts (z/OS Basic Skills)
- DSNTIJMV updates to SYS1.PROCLIB (Db2 Installation and Migration)

**Related tasks:**
- Accessing the correct Db2 program library (Db2 Installation and Migration)
Chapter 7. Monitoring and controlling online utilities

You can monitor utilities, run utilities concurrently, terminate utilities, and restart utilities.

Monitoring utilities

You can check the status of any Db2 online utility jobs. You can also check the phase in which the utility is executing or stopped.

About this task

An online utility can have one of the following statuses:

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>The utility is started.</td>
</tr>
<tr>
<td>Stopped</td>
<td>The utility abnormally stopped before completion, but the table spaces and indexes that were accessed by the utility remain under utility control. To make the data available again, take one of the following actions:</td>
</tr>
<tr>
<td></td>
<td>• Correct the condition that stopped the utility, and restart the utility so that it runs to completion.</td>
</tr>
<tr>
<td></td>
<td>• Terminate the utility with the Db2 TERM UTILITY command.</td>
</tr>
<tr>
<td>Terminated</td>
<td>The Db2 TERM UTILITY command requested that the utility be terminated.</td>
</tr>
</tbody>
</table>

Db2 online utility execution is divided into phases. Each utility starts with the UTILINIT phase, which performs initialization and setup. Each utility finishes with a UTILTERM phase, which cleans up after processing completes. The other phases of online utility execution differ, depending on the utility. For a list of execution phases for each utility, see the documentation for the utility.

Procedure

To monitor utilities:

1. 
2. Use the Db2 DISPLAY UTILITY command. The status and phase is displayed in message DSNU100I, DSNU105I or DSNU106I. (All of these messages are documented under DSNU100I.)

Related tasks:

- “Terminating an online utility” on page 43
- “Restarting an online utility” on page 44

Related reference:

- DISPLAY UTILITY (Db2) (Db2 Commands)
- TERM UTILITY (Db2) (Db2 Commands)

Related information:

- DSNU100I (Db2 Messages)
Monitoring processor use by utilities

You can run traces to collect data about the amount of processor time that is used by online utilities.

Procedure

To monitor processor use by utilities:

Run a trace that writes IFCID 0025 trace records. For a list of trace classes that include IFCID 0025, see and [Performance trace (Db2 Performance)].

The following types of information are collected:

• Whether DFSORT or DB2SORT was invoked.
• The number of parallel data or index sorts that were performed.
• The amounts of time that a utility job used:
  – Total elapsed time
  – Total CPU time
  – Total zIIP time (if an accounting class 1 trace is also activated)
  – CPU time for sorts
  – zIIP time for sorts (if the sort program provided this value)

Related concepts:

(Db2 trace (Db2 Performance))

Related tasks:

Controlling the Db2 trace (Db2 Administration Guide)

Monitoring use of IBM specialty engines (Db2 Performance)

Subsystem parameters for refining DFSMSdss COPY operation with utilities

You can use subsystem parameters to control whether utilities that invoke DFSMSdss COPY use FlashCopy technology.

The following utilities invoke DFDSMSdss COPY:

• CHECK DATA with SHRLEVEL CHANGE
• CHECK INDEX with SHRLEVEL CHANGE
• CHECK LOB with SHRLEVEL CHANGE
• COPY with FLASHCOPY YES or FLASHCOPY CONSISTENT
• LOAD with FLASHCOPY YES or FLASHCOPY CONSISTENT
• REBUILD INDEX with FLASHCOPY YES or FLASHCOPY CONSISTENT
• RECOVER from FlashCopy image copies
• REORG INDEX with FLASHCOPY YES or FLASHCOPY CONSISTENT
• REORG TABLESPACE with FLASHCOPY YES or FLASHCOPY CONSISTENT
You can use the following subsystem parameters to control how FlashCopy technology is used by all utilities that invoke DFSMSdss COPY:

**FLASHCOPY_XRCP**

Specifies whether a data set level FlashCopy operation is allowed to a primary volume in a z Global Mirror (Extended Remote Copy or XRC) relationship when the z/OS DFSMSdss support for RPFC for XRC is installed and enabled.

**FLASHCOPY_PPRC**

Specifies the behavior for DFSMSdss FlashCopy requests when the target disk storage volume is the primary device in a Peer-to-Peer Remote Copy (Metro Mirror) relationship.

You can use the following subsystem parameters to specify how certain utilities use FlashCopy:

**CHECK_FASTREPLICATION**

Specifies whether the CHECK utilities direct DFSMSdss COPY to use FlashCopy as the preferred method for copying objects to shadow data sets, or as the only method for copying objects to shadow data sets.

**REC_FASTREPLICATION**

Specifies how the RECOVER utility directs DFSMSdss COPY to restore an image copy that was created with FlashCopy. REC_FASTREPLICATION directs DFSMSdss COPY to use FlashCopy as the preferred method, as the only method, or not to use FlashCopy.

The parameters that DFSMSdss COPY specifies for the CHECK utilities depend on the combination of values for FLASHCOPY_PPRC and CHECK_FASTREPLICATION that you specify, as shown in the following table.

<table>
<thead>
<tr>
<th>FLASHCOPY_PPRC value</th>
<th>CHECK_FASTREPLICATION value</th>
</tr>
</thead>
<tbody>
<tr>
<td>blank</td>
<td>FASTREP(PREF)</td>
</tr>
<tr>
<td>NONE</td>
<td>FASTREP(PREF)</td>
</tr>
<tr>
<td>PREFERRED</td>
<td>FASTREP(PREF)</td>
</tr>
<tr>
<td>REQUIRED</td>
<td>FASTREP(PREF)</td>
</tr>
</tbody>
</table>

The parameters that DFSMSdss COPY specifies for the RECOVER utility depend on the combination of values for FLASHCOPY_PPRC and REC_FASTREPLICATION that you specify, as shown in the following table.

<table>
<thead>
<tr>
<th>FLASHCOPY_PPRC value</th>
<th>REC_FASTREPLICATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>blank</td>
<td>FASTREP(NONE)</td>
</tr>
<tr>
<td>PREFERRED</td>
<td>FASTREP(PREF)</td>
</tr>
<tr>
<td>REQUIRED</td>
<td>FASTREP(REQ)</td>
</tr>
</tbody>
</table>
Table 3. FLASHCOPY_PPRC and REC_FASTREPLICATION values and resulting DFSMSdss COPY parameter values for the RECOVER utility (continued)

<table>
<thead>
<tr>
<th>FLASHCOPY_PPRC value</th>
<th>REC_FASTREPLICATION</th>
<th>PREFERRED</th>
<th>REQUIRED</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>FASTREP(NONE)</td>
<td>FASTREP(PREF)</td>
<td>FASTREP(REQ)</td>
</tr>
<tr>
<td></td>
<td>FCTOPPRCP(PresMirNone)</td>
<td>FCTOPPRCP(PresMirNone)</td>
<td></td>
</tr>
<tr>
<td>PREFERRED</td>
<td>FASTREP(NONE)</td>
<td>FASTREP(PREF)</td>
<td>FASTREP(REQ)</td>
</tr>
<tr>
<td></td>
<td>FCTOPPRCP(PresMirPref)</td>
<td>FCTOPPRCP(PresMirPref)</td>
<td></td>
</tr>
<tr>
<td>REQUIRED</td>
<td>FASTREP(NONE)</td>
<td>FASTREP(PREF)</td>
<td>FASTREP(REQ)</td>
</tr>
<tr>
<td></td>
<td>FCTOPPRCP(PresMirReq)</td>
<td>FCTOPPRCP(PresMirReq)</td>
<td></td>
</tr>
</tbody>
</table>

Related concepts:
“FlashCopy image copies” on page 157

Determining why a utility failed to complete

If an online utility job completes normally, it issues return code 0. If it completes with warning messages, it issues return code 4. Return code 8 means that the job failed to complete. Return code 12 means that an authorization error occurred.

Procedure

To determine why a utility failed to complete:

Use the following list of possible scenarios to help resolve the problem:

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Db2 terminates the utility job step and any subsequent utility steps.</td>
<td>Submit a new utility job to execute the terminated steps. Use the same utility identifier for the new job to ensure that no duplicate utility job is running.</td>
</tr>
<tr>
<td>Db2 does not execute the particular utility function, but prior utility functions are executed.</td>
<td>Submit a new utility step to execute the function.</td>
</tr>
<tr>
<td>Db2 places the utility function in the stopped state.</td>
<td>Restart the utility job step at either the last commit point or the beginning of the phase by using the same utility identifier. Alternatively, use the TERM UTILITY command to terminate the job step and then resubmit it.</td>
</tr>
</tbody>
</table>

Related information:
“Restarting an online utility” on page 44
“Terminating an online utility” on page 43
Db2 terminates the utility and issues return code 8.

One or more objects might be in a restrictive or advisory state. Resolve that status and resubmit the job.

Also, a DEADLINE condition in REORG might have terminated the reorganization.

Related information:
- Appendix B, “Advisory or restrictive states,” on page 1037
- “Syntax and options of the REORG INDEX control statement” on page 516
- “Syntax and options of the REORG TABLESPACE control statement” on page 556

### Terminating an online utility

You can terminate the execution of an active utility or release the resources that are associated with a stopped utility.

**About this task**

After you terminate a utility, you cannot restart the terminated utility job. The objects on which the utility was operating might be left in an indeterminate state. In many cases, you cannot rerun the same utility without first recovering the objects on which the utility was operating. The situation varies, depending on the utility and the phase that was in progress when you issued the command. These considerations about the state of the object are particularly important when you terminate the COPY, LOAD, and REORG utilities.

**Procedure**

To terminate an online utility:

Issue the **TERM UTILITY** command. If the utility was started in a previous release of Db2, issue the **TERM UTILITY** command from that release.

In a data sharing environment, the following rules apply:

- For active utilities, issue the command from the Db2 subsystem on which the utility was started. If a Db2 subsystem fails while a utility is in progress, you must restart that Db2 subsystem, and then you can terminate the utility from any system.
- For stopped utilities, you can terminate them from any active member of the data sharing group.
- In a data sharing coexistence environment, you can terminate a utility only on the same release in which the utility was started.

If the utility is active, Db2 terminates the execution of the utility at the next commit point and then performs any necessary cleanup operations. If the utility is stopped, Db2 releases the resources that are associated with the stopped utility.
Example of conditionally terminating a utility

You might choose to put the `TERM UTILITY` command in a conditionally executed job step; for example, if you never want to restart certain utility jobs. The following figure shows such an example.

```bash
//TERM EXECPGM=IKJEFT01,COND=((8,GT,S1),EVEN)
//*
//*******************************************************************************/
//* IF THE PREVIOUS UTILITY STEP, S1, ABENDS, ISSUE A
//* TERMINATE COMMAND. IT CANNOT BE RESTARTED.
//*******************************************************************************/
//*
//SYSPRINT DD SYSOUT=A
//SYSTSPRT DD SYSOUT=A
//SYSOUT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//SYSIN DD *
DSN SYSTEM(ASN)
-TERM UTILITY(TEMP)
END
/*
Figure 5. Example of conditionally executed TERM UTILITY

Alternatively, consider specifying the TIMEOUT TERM parameter for some online REORG situations.

Related reference:

-TERM UTILITY (Db2) (Db2 Commands)
“Syntax and options of the REORG INDEX control statement” on page 516
“Syntax and options of the REORG TABLESPACE control statement” on page 556

Restarting an online utility

If a utility finishes abnormally, you might be able to restart it. You need only resubmit the job and Db2 attempts to restart it. However, you might need or choose to make other changes before you resubmit the job.

About this task

Utilities can be restarted with one of the following two methods:

Phase restart
The utility restarts from the beginning of the phase that was being processed. This method is indicated by the parameter `RESTART(PHASE)`.

Current restart
The utility restarts from the last checkpoint that was taken during the execution of the utility phase. If the utility phase does not take any checkpoints or has not reached the first checkpoint, current restart is equivalent to phase restart. This method is indicated by the parameter `RESTART` or `RESTART(CURRENT)`.

Restarting a utility job is not always possible. If you cannot restart a utility job, you might have to terminate it to make the data available to other applications. To
terminate a utility job, issue the Db2 TERM UTILITY command. For more information, see “Terminating an online utility” on page 43.

Important: In data sharing release coexistence, you must restart a utility on a member on the same Db2 release as the member where the utility job was originally submitted. The same utility ID (UID) must be used to restart the utility. That UID is unique within a data sharing group. However, if Db2 fails, you must restart Db2 on either the same or another z/OS system before you restart the utility.

Procedure

To restart an online utility:
1. Correct the problem that caused the utility job to stop.
2. If needed, make changes to the utility job. Use the following guidelines:
   • Do not specify z/OS automatic step restart.
   • Ensure that the required data sets are properly defined.
     Recommendation: Allocate the data sets by using TEMPLATE statements that do not specify the DISP and SPACE parameter values. When these parameters are not specified, Db2 determines the correct disposition and size of these data sets.
   • Check the DISP parameters on the DD statements. For example, for DD statements that have DISP=NEW and need to be reused, change DISP to OLD or MOD. If generation data groups (GDGs) are used and any (+1) generations were cataloged, ensure that the JCL is changed to GDG (+0) for such data sets.
     Automatically generated JCL normally has DISP=MOD. DISP=MOD allows a data set to be allocated during the first execution and then reused during a restart.
     When restarting a job that involves templates, Db2 automatically changes the disposition from NEW to MOD. Therefore, you do not need to change template specifications for restart.
   • For any data sets that are not dynamically allocated, ensure that the DD name that is specified in the restart JCL matches the DD name for the original job. Do not change DD names on a restart job.
     If the LOAD utility is restarted with a different SYSREC data set from the SYSREC data set for the initial invocation, the data control block (DCB) parameters must be the same as the SYSREC DCB parameters for the initial invocation. START(PHASE) is recommended. START(CURRENT) might have unpredictable results because data set checkpoint information is not used during restart processing.
   • For any data sets that are dynamically allocated, ensure that the file sequence numbers in the restart job match those file sequence numbers in the original job.
   • For data sets that are not cataloged, ensure that any explicit VOLSER specifications in the restart job match those specifications in the original job.
     If you copy a work data set, such as SYSUT1, after abend B37, and the number of volumes changes, do not specify START(CURRENT). If you do, abend 413-1C occurs. To prevent this abend, start the utility in START(PHASE).
   • For any cataloged data sets, do not specify VOLSER. Let Db2 determine the VOLSER value of the data sets from the system catalog.
Include all of the original utility statements and do not change the position of any other utility statements that have executed. During restart processing, Db2 remembers the relative position of the stopped utility statement in the input stream. The exception is DIAGNOSE statements. You can add or delete DIAGNOSE statements as needed.

Do not change the utility function that is stopped and the Db2 objects on which it is operating. However, you can change other parameters that are related to the stopped step and to subsequent utility steps.

Do not change any EXEC SQL or OPTIONS utility control statements that executed before the stopped utility, if possible. If you must change these utility control statements, use caution; any changes can cause the restart processing to fail. For example, if you specify a valid OPTIONS statement in the initial invocation, and then on restart, specify OPTIONS PREVIEW, the restart fails.

Use caution if you change LISTDEF lists before you restart the utility. When Db2 restarts list processing, it uses a saved copy of the list. Modifying the LISTDEF list that is referenced by the stopped utility has no effect. Only control statements that follow the stopped utility are affected. See “How utilities restart with lists” on page 49.

Use caution if you change TEMPLATE statements before restarting the utility. See “How utilities restart with templates” on page 49.

If you are using the DSNUTILS stored procedure, specify NONE or ANY for the utility-name parameter. These values suppress the dynamic allocation that is normally performed by DSNUTILS. You can then specify TEMPLATE statements (in the utstmt parameter) to allocate the necessary data sets.

3. Resubmit the job with the same utility ID and optionally specify a RESTART parameter value.
Db2 recognizes the utility ID and restarts the utility job if possible. Db2 retrieves information about the stopped utility from the SYSIBM.SYSUTIL directory table.

For each utility, Db2 uses the default RESTART value that is specified in the following table.

<table>
<thead>
<tr>
<th>Utility</th>
<th>Default RESTART value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACKUP SYSTEM</td>
<td>RESTART(CURRENT)</td>
</tr>
<tr>
<td>CATENFM</td>
<td>No restart</td>
</tr>
<tr>
<td>CATMAINT</td>
<td>No restart</td>
</tr>
<tr>
<td>CHECK DATA</td>
<td>RESTART(CURRENT)</td>
</tr>
<tr>
<td>CHECK INDEX</td>
<td>RESTART(CURRENT)</td>
</tr>
<tr>
<td>CHECK LOB</td>
<td>RESTART(CURRENT)</td>
</tr>
<tr>
<td>COPY</td>
<td>RESTART(CURRENT)</td>
</tr>
<tr>
<td>COPYTOCOPY</td>
<td>RESTART(CURRENT)</td>
</tr>
<tr>
<td>DIAGNOSE</td>
<td>Restarts from the beginning</td>
</tr>
<tr>
<td>EXEC SQL</td>
<td>Restarts from the beginning</td>
</tr>
<tr>
<td>LISTDEF</td>
<td>Restarts from the beginning</td>
</tr>
<tr>
<td>LOAD</td>
<td>RESTART(CURRENT) or RESTART(PHASE)*</td>
</tr>
<tr>
<td>MERGECOPY</td>
<td>RESTART(PHASE)</td>
</tr>
<tr>
<td>MODIFY RECOVERY</td>
<td>RESTART(CURRENT)</td>
</tr>
<tr>
<td>Utility</td>
<td>Default RESTART value</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>MODIFY STATISTICS</td>
<td>RESTART(CURRENT)</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>Restarts from the beginning</td>
</tr>
<tr>
<td>QUIESCE</td>
<td>RESTART(CURRENT)</td>
</tr>
<tr>
<td>REBUILD INDEX</td>
<td>RESTART(PHASE)</td>
</tr>
<tr>
<td>RECOVER</td>
<td>RESTART(CURRENT)</td>
</tr>
<tr>
<td>REORG INDEX</td>
<td>RESTART(CURRENT) or RESTART(PHASE)</td>
</tr>
<tr>
<td>REORG TABLESPACE</td>
<td>RESTART(CURRENT) or RESTART(PHASE)</td>
</tr>
<tr>
<td>REPAIR</td>
<td>No restart</td>
</tr>
<tr>
<td>REPORT</td>
<td>RESTART(CURRENT)</td>
</tr>
<tr>
<td>RESTORE SYSTEM</td>
<td>RESTART(CURRENT)</td>
</tr>
<tr>
<td>RUNSTATS</td>
<td>RESTART(CURRENT)</td>
</tr>
<tr>
<td>STOSPACE</td>
<td>RESTART(CURRENT)</td>
</tr>
<tr>
<td>TEMPLATE</td>
<td>Restarts from the beginning</td>
</tr>
<tr>
<td>UNLOAD</td>
<td>RESTART(CURRENT)</td>
</tr>
</tbody>
</table>

**Note:**
1. The RESTART value that Db2 uses for these utilities depends on the situation. For a complete explanation for each utility, see the information in the third column.

You can override the default RESTART value by specifying the RESTART parameter in the original JCL data set. See “Overriding the default utility restart behavior by using the RESTART parameter.”

If a utility is restarted in the UTILINIT phase, it is re-executed from the beginning of the phase.

**What to do next**

If you restarted a LOAD, REBUILD INDEX, or REORG job with the STATISTICS option, run the RUNSTATS utility after the restarted job completes. When you restart these jobs, Db2 does not collect inline statistics. The exception is REORG UNLOAD PAUSE; when restarted after the pause, REORG UNLOAD PAUSE collects statistics.

**Related tasks:**

“Terminating an online utility” on page 43

**Related information:**

-[Automatic Step Restart (DFSMSdfp Checkpoint/Restart)]

**Overriding the default utility restart behavior by using the RESTART parameter**

You do not need to use the RESTART parameter to restart a utility job.

When you resubmit a job that finished abnormally and has not been terminated, Db2 recognizes the utility ID from the SYSIBM.SYSUTIL directory table and restarts the utility job if possible. However, if you want to override the default RESTART value, you can update the original JCL data set by adding the RESTART parameter.
parameter. Any RESTART values that you specify always override the default values. Db2 ignores the RESTART parameter if you are submitting the utility job for the first time.

**Procedure**

To override the default utility restart behavior, add the RESTART parameter by using one of the following methods:

- If you invoke the utility by using DB2I, specify the RESTART value on a DB2I panel:
  1. Access the DB2 Utilities panel.
  2. Complete the panel fields.
  3. Change field 5 (RESTART) to CURRENT or PHASE, depending on the restart method that you want.
  4. Press Enter.

  For information about the DB2 Utilities panel fields, see “Invoking an online utility by using the DB2 Utilities panel in DB2I” on page 21.

- If you invoke the utility by using the DSNU CLIST command, change the value of the RESTART parameter on the command. See “DSNU CLIST command syntax” on page 27.

- If you created your own JCL to invoke the utility, specify RESTART (CURRENT) or RESTART(PHASE) on the EXEC statement. For information about the RESTART parameter, see “Invoking a Db2 online utility by creating the JCL dataset yourself” on page 36.

**Restarting a utility after the output data set is full**

If a utility job terminates with an out-of-space condition on the output data set, you might need to restart the job at the last commit point.

**Procedure**

To restart a utility after the output data set is full:

1. Copy the output data set to a temporary data set. Use the same data control block (DCB) parameters. Use z/OS utilities that do not reblock the data set during the copy operation (for example, DFSMSdss ADDRDSU or DFSORT ICEGENER). Avoid using the IEBGENER or ISPF 3.3 utilities.
2. Delete or rename the output data set. Ensure that you know the current DCB parameters.
3. Redefine the output data set with additional space. Use the same VOLSER (if the data set is not cataloged), the same DSNNAME, and the same DCB parameters.
4. Copy the data from the temporary data set into the new, larger output data set. Use z/OS utilities that do not reblock the data set during the copy operation (for example, DFSMSdss ADDRDSU or DFSORT ICEGENER).
5. **Restart the online utility**

**Related concepts:**

- Examples of Data Set Copy Operations (z/OS DFSMSdss Storage Administration)

**Related reference:**

- DCB Subparameters (MVS JCL Reference)
How utilities restart with templates

Unlike most other utility control statements, TEMPLATE utility control statements can be modified before you restart a utility. In some cases, they must be modified to correct a prior failure.

If possible, do not change the TEMPLATE control statement. If you must change it, use caution. In some cases, modifications can cause restart processing to fail. For example, if you change the template name of a temporary work data set that was opened in an earlier phase and closed but is to be used later, restart processing fails.

Restriction: When a TEMPLATE utility control statement includes the PATH keyword, the utility that uses that template cannot be restarted.

When a utility is restarted, template allocation automatically adjusts data set dispositions to reallocate the data sets from the prior execution. You do not need to change the TEMPLATE DISP option. Db2 also takes checkpoints for the values that are used for TEMPLATE DSN variables, and the old values are reused when the utility is restarted.

If the failure of the utility job was due to space problems on a data set, the same restart considerations apply as if DD statements were used.

If the failure of the utility job was due to insufficient space on a volume, you can alter the TEMPLATE statement. How the TEMPLATE statement needs to be altered depends on whether the SPACE keyword was specified. If SPACE was specified, specify a different volume or alter the primary and secondary space quantities. If SPACE was not specified, specify a different volume or add the PCTPRIME and NBRSECND options. Lower the value of the PCTPRIME option to decrease the size of the primary allocation, and increase the value of the NBRSECND option to decrease the size of the secondary allocation.

Related tasks:
“Restarting a utility after the output data set is full” on page 48

Related reference:
Chapter 34, “TEMPLATE,” on page 793

How utilities restart with lists

Lists are defined by the LISTDEF utility. Unlike other utility control statements, LISTDEF control statements can be modified before you restart a utility. However, those changes do not affect the currently running utility. The changed list affects only those utility control statements that follow the stopped utility.

When you originally submit a utility control statement that references a list, Db2 expands the contents of the list and saves the list before executing the utility. Db2 uses this saved list to restart the utility at the point of failure. After LISTDEF repositions in the list at the point of failure, individual utility restart processing is invoked. This restart behavior varies by utility. After the utility is successfully restarted, the LISTDEF list is re-expanded before it is used by subsequent utilities in the same job step.
To determine whether the utility that you are restarting is processing a list or the size of the list that the utility is processing, issue the DISPLAY UTILITY command. If a list is being used, the size is reported in message DSNU100 or DSNU105 in the DISPLAY UTILITY output.

**Related reference:**
- Chapter 18, “LISTDEF,” on page 215

**Related information:**
- DSNU100I (Db2 Messages)
Chapter 8. BACKUP SYSTEM

The online BACKUP SYSTEM utility invokes z/OS DFSMShsm to copy the volumes on which the Db2 data and log information resides. These system-level backups can be taken for either a Db2 subsystem or data sharing group. You can later run the RESTORE SYSTEM utility to recover the subsystem or data sharing group.

You can use BACKUP SYSTEM to copy all data for a single application (for example, when Db2 is the database server for a resource planning solution).

Restrictions:

• All data sets that you want to copy must be SMS-managed data sets.
• A data sharing environment, no failed abnormally quiesced members can exist. Otherwise, the BACKUP SYSTEM request fails.

The BACKUP SYSTEM utility uses copy pools. You must define these copy pools before you run BACKUP SYSTEM. See “Copy pools” on page 54.

Output

The output for BACKUP SYSTEM is the copy of the volumes on which the Db2 data and log information resides. The copies are called system-level backups.

The BACKUP SYSTEM history is recorded in the bootstrap data sets (BSDSs).

BACKUP SYSTEM does not reset COPY-pending status. To remove COPY-pending status, take an image copy of the affected object.

Related information:

“COPY-pending status” on page 1040

Authorization required

To execute this utility, you must use a privilege set that includes SYSCTRL or SYSADM authority.

Execution phases of BACKUP SYSTEM

The BACKUP SYSTEM utility operates in these phases:

Phase   Description
UTILINIT   Performs initialization and setup.
COPY     Copies data.
UTILTERM  Performs cleanup.

Related concepts:

Point-in-time recovery with system-level backups (Db2 Administration Guide)
Syntax and options of the BACKUP SYSTEM control statement

The BACKUP SYSTEM utility control statement, with its multiple options, defines the function that the utility job performs.

Use the ISPF/PDF edit function to create a control statement and to save it in a sequential or partitioned data set. When you create the JCL for running the job, use the SYSIN DD statement to specify the name of the data set that contains the utility control statement.

When you specify BACKUP SYSTEM, you can specify only the following statements in the same step:

- DIAGNOSE
- OPTIONS PREVIEW
- OPTIONS OFF
- OPTIONS KEY
- OPTIONS EVENT WARNING

In addition, BACKUP SYSTEM must be the last statement in SYSIN.

Syntax diagram

```
BACKUP SYSTEM
  FULL
  DATA ONLY
  ESTABLISH FCIINCREMENTAL
  END FCIINCREMENTAL
  FORCE
  DUMP
  DUMPONLY
    dumpclass-spec
    FORCE
    TOKEN—(X'byte-string')
    dumpclass-spec

dumpclass-spec:

DUMPCLASS (dc1, dc2, dc3, dc4, dc5)
```
Option descriptions

FULL
Indicates that you want to copy both the database copy pool and the log copy pool.

You must ensure that the database copy pool is set up to contain the volumes for the databases and the associated integrated catalog facility (ICF) catalogs. You must also ensure that the log copy pool is set up to contain the volumes for the BSDSs, the active logs, and the associated catalogs.

Use BACKUP SYSTEM FULL to allow for recovery of both data and logs. You can use the RESTORE SYSTEM utility to recover the data. However, RESTORE SYSTEM does not restore the logs; the utility only applies the logs. If you want to restore the logs, you must use another method to restore them.

DATA ONLY
Indicates that you want to copy only the database copy pool. You must ensure that the database copy pool is set up to contain the volumes for the databases and the associated ICF catalogs.

ESTABLISH FCINCREMENTAL
Specifies that a persistent incremental FlashCopy relationship is to be established, if none exists, for source copy volumes in the database copy pool. Use this keyword once to establish the persistent incremental FlashCopy relationships. Subsequent invocations of BACKUP SYSTEM (without this keyword) automatically process the persistent incremental FlashCopy relationship.

END FCINCREMENTAL
Specifies that a last incremental FlashCopy is to be taken and that the persistent incremental FlashCopy relationship is to be withdrawn for all of the volumes in the database copy pool. Use this keyword only if you do not want further incremental FlashCopy backups of the database copy pool.

FORCE
Indicates that you want to overwrite the oldest DFSMSshm version of the fast replication copy of the database copy pool. You can overwrite these copy pools even if the dump to tape or the copy pool's DFSMSshm dump classes were initiated, but are only partially completed.

Use the FORCE option only if it is more important to take a new system-level backup than to save a previous system-level backup to tape.

DUMP
Indicates that you want to create a fast replication copy of the database copy pool and the log copy pool on disk and then initiate a dump to tape of the fast replication copy. The dump to tape begins after Db2 successfully establishes relationships for the fast replication copy.

The BACKUP SYSTEM utility does not wait for the dump processing to complete.

This option requires z/OS Version 1.8.

DUMPCCLASS
Indicates the DFSMSshm dump class that you want to use for the dump processing. You can specify up to five dump classes. If you do not specify a dump class, Db2 uses the default dump classes that are defined for the copy pools.
**DUMPONLY**
Indicates that you want to create a dump on tape of an existing fast replication copy (that resides on the disk) of the database copy pool and the log copy pool. You can also use this option to resume a dump process that failed.

The BACKUP SYSTEM utility does not wait for the dump processing to complete.

This option requires z/OS Version 1.8.

**TOKEN (X'byte-string')**
Specifies which fast replication copy of the database copy pool and the log copy pool to dump to tape.

The token is a 36 digit or 44-digit hexadecimal byte string that uniquely identifies each system-level backup and is reported in the DSNJU0004 job output. For a data sharing system, run DSNJU0004 with the MEMBER option so that the system-level backup information is displayed for all members.

Backups that are taken before the BSDS is converted to 10-byte extended format are identified with 36-digit tokens. Backups that are taken after the BSDS is converted to 10-byte extended format are identified with 44-digit tokens. If specified, the token must be in the correct format for the system-level backup.

If you do not specify TOKEN, the most recent fast replication copy of the copy pools is dumped to tape.

---

**Before running BACKUP SYSTEM**

Certain activities might be required before you run the BACKUP SYSTEM utility, depending on your situation.

To run BACKUP SYSTEM, ensure that the following conditions are true:

- The data sets that you want to copy are SMS-managed data sets.
- You have disk control units that support ESS FlashCopy.
- Any needed copy pools are defined. See "Copy pools."
- The ICF catalog for the data must be on a separate volume than the ICF catalog for the logs.
- An SMS backup storage group is defined for each storage group in the copy pools.

**Related information:**

- Defining Copy Pools (DFSM Sao Storage Administration)
- Defining Storage Group Attributes (DFSM Sao Storage Administration)
- Altering copy pools (DFSM Sao Storage Administration)

**Copy pools**

The BACKUP SYSTEM utility uses copy pools. A *copy pool* is a defined set of storage groups that contain data that DFSMShsm can back up and recover collectively. Each of these storage groups contains the name of the associated backup storage group for storing the backup.

Each Db2 subsystem can have up to two copy pools, one for databases and one for logs. BACKUP SYSTEM copies the volumes that are associated with these copy pools at the time of the copy.
You must define any needed copy pools before you run BACKUP SYSTEM. You must at least define a database copy pool, and that definition must include the ICF catalog names. (You can add the ICF catalog names to the database copy pool definition by altering the copy pools.) If you plan to also copy the logs, define another copy pool for your logs.

Use the following Db2 naming convention when you define the required copy pools:

\[ DSN\$locn-name\$cp-type \]

where:

- **DSN**: The unique Db2 product identifier.
- **\$$**: A delimiter. You must use the dollar sign character ($$).
- **locn-name**: The Db2 location name.
- **cp-type**: The copy pool type. Use DB for database and LG for log.

If you use Metro Mirror (Peer-to-Peer Remote Copy or PPRC) or z Global Mirror (Extended Remote Copy or XRC), and Remote Pair FlashCopy (RPFC) support is enabled, use the following fields on the Copy Pool Define panel to allow volume-level FlashCopy operations on the primary volumes by DFSMSHsm:

- FRBACKUP to PPRC Primary Volumes allowed
- FRRECOV to PPRC Primary Volumes allowed
- FRBACKUP to XRC Primary Volumes allowed
- FRRECOV to XRC Primary Volumes allowed

**Related information:**

- [Steps for defining a copy pool (DFSMSdftp Storage Administration)](#)
- [Defining Copy Pools (DFSMSdftp Storage Administration)](#)
- [Defining Storage Group Attributes (DFSMSdftp Storage Administration)](#)

### Data sets that BACKUP SYSTEM uses

The BACKUP SYSTEM utility uses a number of data sets during its operation.

The following table lists the data sets that the BACKUP SYSTEM utility uses. The table lists the DD name that is used to identify the data set, a description of the data set, and an indication of whether it is required. Include statements in your JCL for each required data set.

<table>
<thead>
<tr>
<th>Data sets</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSIN</td>
<td>An input data set that contains the utility control statement</td>
<td>Yes</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>An output data set for messages</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Related concepts:**

- [“Data sets that online utilities use” on page 16](#)
Concurrent and compatibility for BACKUP SYSTEM

The BACKUP SYSTEM utility has certain concurrency and compatibility characteristics associated with it.

The BACKUP SYSTEM utility can run concurrently with any other utility; however, it must wait for the following Db2 events to complete before the copy can begin:

- Extending of data sets
- Writing of 32-KB pages
- Writing close page set control log records (PSCRs)
- Creating data sets (for table spaces, indexes, and so forth)
- Deleting data sets (for dropping tables spaces, indexes, and so forth)
- Renaming data sets (for online reorganizing of table spaces, indexes, and so forth during the SWITCH phase)

Only one BACKUP SYSTEM job can be running at one time.

BACKUP SYSTEM cannot run concurrently with utilities that use FlashCopy to create data sets in the database copy pool. For example, suppose that CHECK INDEX SHRLEVEL CHANGE does a FlashCopy from a source object to a shadow data set. The disk volume where the shadow data set resides becomes the target in a FlashCopy relationship. If this disk volume is in the database copy pool, BACKUP SYSTEM cannot copy it.

For the CHECK INDEX, CHECK DATA, and CHECK LOB utilities, you can use subsystem parameter UTIL_TEMP_STORCLAS to specify an alternative storage class that contains volumes that are not in the database copy pool. When UTIL_TEMP_STORCLAS is specified, the CHECK utilities use the alternative storage class to create the shadow data sets. Therefore, volumes that are targets in a FlashCopy relationship after the CHECK utilities run are not in the database copy pool.

Dumping a fast replication copy to tape

With the BACKUP SYSTEM online utility, you can dump a fast replication copy of a system-level backup to tape. You can then manage the available disk space, retain the system-level backups, and provide a means of recovery after a media failure.

Procedure

To dump a fast replication copy of a system-level backup to tape that was taken without the DUMP option, or to re-initiate dump processing that has failed:

1. Identify the token (a 36 digit or 44 digit hexadecimal byte string) in the DSNJU004 output.
2. Create and run your utility control statement with the DUMPONLY option. Specify the token if the system-level backup is not the most recent system-level backup taken.

Restriction: Do not dump system-backups to the same tape that contains image copies or concurrent copies because the RECOVER utility requires access to both
3. Run the DFSMShsm command LIST COPYPOOL with the ALLVOLS option to verify that the dump to tape was successful. The BACKUP SYSTEM utility issues the DFSMShsm command to initiate a dump, but it does not wait for the dump to be completed.

---

**Backups of log copy pools**

If you take backups of both the log and database copy pool, you can use the backups to restore the log copy pool.

When you use backups to restore the log copy pool, if the active log data sets are stripped, or if the log copy pool is for a data sharing environment, you must specify the data complete LRSN during the conditional restart in the following scenarios:

- You are cloning a Db2 system by using a system-level backup as the source. In this case, conditionally restart Db2 with an ENDRBA or ENDLRSN that is equal to the data complete LRSN of the system-level backup.
- You are performing a system-level point-in-time recovery. In this case, conditionally restart Db2 with the log truncation point equal to or less than the data complete LRSN of the system-level backup. Use the data complete LRSN as the CRESTART ENDRBA, ENDLRSN, or SYSPITR log truncation point.

You can determine the data complete LRSN from the following places:

- Message DSNU1614I, which is generated when BACKUP SYSTEM completes successfully
- The report that is generated by the print log map utility (DSNJU004)

**Related concepts:**

- “Before running RESTORE SYSTEM” on page 734
- Conditional restart with system-level backups (Db2 Administration Guide)

**Related reference:**

- Chapter 41, “DSNJU004 (print log map),” on page 927

**Related information:**

- DSNU1614I (Db2 Messages)

---

**Termination or restart of BACKUP SYSTEM**

You can terminate BACKUP SYSTEM by using the TERM UTILITY command. BACKUP SYSTEM checks for the TERM UTILITY command before the call to copy data. TERM UTILITY deletes the copy that is being created through the BACKUP SYSTEM utility.

To use TERM UTILITY to terminate BACKUP SYSTEM on a data sharing group, you must issue the command from the member on which the BACKUP SYSTEM utility is invoked.

You can restart a BACKUP SYSTEM utility job, but it starts from the beginning again.

---

**Sample BACKUP SYSTEM control statements**

Use sample control statements as models for developing your own BACKUP SYSTEM utility control statements.
Example 1: Creating a full backup of a Db2 subsystem or data sharing group

The following control statement specifies that the BACKUP SYSTEM utility is to create a full backup copy of a Db2 subsystem or data sharing group. The full backup includes copies of both the database copy pool and the log copy pool. In this control statement, the FULL option is not explicitly specified, because it is the default.

```sql
//STEP1 EXEC DSNUPROC,TIME=1440,
// UTPROC='',
// SYSTEM='DSN'
//SYSIN DD *
      BACKUP SYSTEM
/*
```

Example 2: Creating a data-only backup of a Db2 subsystem or data sharing group

The following control statement specifies that BACKUP SYSTEM is to create a backup copy of only the database copy pool for a Db2 subsystem or data sharing group.

```sql
//STEP1 EXEC DSNUPROC,TIME=1440,
// UTPROC='',
// SYSTEM='DSN'
//SYSIN DD *
      BACKUP SYSTEM DATA ONLY
/*
```

Example 3: Creating a fast replication copy of the database copy pool and dumping the copy to tape

The following control statement specifies that BACKUP SYSTEM is to create a fast replication copy of the database copy pool and initiate a dump to tape of the fast replication copy.

```sql
//SYSOPRB JOB (ACCOUNT), 'NAME', CLASS=K
//UTIL EXEC DSNUPROC, SYSTEM=V91A, UID='TEMB', UTPROC=''
/*
//*
//DSNUPROC.SYSUT1 DD DSN=SYSOPR.SYSUT1,
// DISP=(MOD,DELETE,CATLG),
// SPACE=(16384,(20,20),,,ROUND),
// UNIT=SYSDA
//DSNUPROC.SYSIN DD *
      BACKUP SYSTEM DATA ONLY DUMP
/*
```

Example 4: Creating a fast replication copy of the database copy pool, dumping the copy to tape, and allowing oldest copy to be overwritten

The following control statement specifies that BACKUP SYSTEM is to perform the following actions:

- Create a fast replication copy of the database copy pool
- Initiate a dump to tape of the fast replication copy
- Allow the oldest fast replication copy to be overwritten

```sql
//SYSOPRB JOB (ACCOUNT), 'NAME', CLASS=K
//UTIL EXEC DSNUPROC, SYSTEM=V91A, UID='TEMB', UTPROC=''
/*
```
Example 5: Dumping an existing fast replication copy to tape

The following control statement specifies that BACKUP SYSTEM is to dump the existing fast replication copy X'E5F9F1C1BD1909683AA8A1A600000E6962DE' to tape, using the DB2STGD2 dump class.

```sql
//SYSOPRB JOB (ACCOUNT), 'NAME', CLASS=K
//UTIL EXEC DSNUPROC, SYSTEM=V91A, UID='TEMB', UTPROC= ''
//SYSOPRB JOB (ACCOUNT), 'NAME', CLASS=K
//UTIL EXEC DSNUPROC, SYSTEM=V91A, UID='TEMB', UTPROC= ''
```

```
//DSNUPROC.SYSUT1 DD DSN=SYSOPR.SYSUT1,
// DISP=(MOD,DELETE,CATLG),
// SPACE=(16384,(20,20),,,ROUND),
// UNIT=SYSDA
//DSNUPROC.SYSIN DD *
// BACKUP SYSTEM DATA ONLY DUMP FORCE
/*

Example 5: Dumping an existing fast replication copy to tape

The following control statement specifies that BACKUP SYSTEM is to dump the existing fast replication copy X'E5F9F1C1BD1909683AA8A1A600000E6962DE' to tape, using the DB2STGD2 dump class.

```
//SYSOPRB JOB (ACCOUNT), 'NAME', CLASS=K
//UTIL EXEC DSNUPROC, SYSTEM=V91A, UID= 'TEMB', UTPROC= ''
//SYSOPRB JOB (ACCOUNT), 'NAME', CLASS=K
//UTIL EXEC DSNUPROC, SYSTEM=V91A, UID='TEMB', UTPROC= ''
```

```
//DSNUPROC.SYSUT1 DD DSN=SYSOPR.SYSUT1,
// DISP=(MOD,DELETE,CATLG),
// SPACE=(16384,(20,20),,,ROUND),
// UNIT=SYSDA
//DSNUPROC.SYSIN DD *
BACKUP SYSTEM DATA ONLY DUMPONLY
 TOKEN X'E5F9F1C1BD1909683AA8A1A600000E6962DE'
 DUMPCLASS(DB2STGD2)
/*
```
Chapter 9. CATENFM

The CATENFM online utility enables a Db2 subsystem to enter Db2 Db2 11 enabling-new-function mode and Db2 11 new-function mode. It also enables a Db2 subsystem to return to enabling-new-function mode from new-function mode.

All new Db2 11 functions are unavailable when the subsystem is in conversion mode or enabling-new-function mode.

Output

Output from the CATENFM utility consists of:

- If you specify the CONVERT option, the CATENFM utility converts table spaces during the enabling-new-function mode process.
- If you specify the ALTER option, some objects in the Db2 catalog are altered or created.
- For other options, there is no output.

Authorization required

The required authorization for CATENFM is installation SYSADM.

Execution phases of CATENFM

The CATENFM utility operates in these phases:

<table>
<thead>
<tr>
<th>Phase</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTILINIT</td>
<td>Performs initialization and setup</td>
</tr>
<tr>
<td>UTILTERM</td>
<td>Performs cleanup</td>
</tr>
</tbody>
</table>

Syntax and options of the CATENFM control statement

The CATENFM utility control statement, with its multiple options, defines the function that the utility job performs.

The CATENFM utility is invoked by jobs DSNTIJEN, DSNTIJNF, DSNTIJES, and DSNTIJCS.

Syntax diagram

```
CATENFM [START] [COMPLETE] [ENFMON] [CMON] [CONVERT=INPUT=table-space-name]
```
Option descriptions

START
Invokes the CATENFM utility and indicates the start of enabling-new-function mode processing. No start processing is done if CATENFM START was run previously; however, you can run the CATENFM utility as many times as needed.

COMPLETE
Checks if the Db2 subsystem has completed enabling-new-function mode processing. If the subsystem has completed this processing, the CATENFM utility returns 0, and the subsystem enters new-function mode.

ENFMON
Returns Db2 to enabling-new-function mode. If the subsystem is currently in enabling-new-function mode, no change occurs. If the subsystem has been to new-function mode, ENFMON returns it to enabling-new-function* mode. New Db2 11 functions are not available in Db2 11 enabling-new-function mode.

CMON
Returns Db2 to conversion mode. If the system is currently in conversion mode, no change occurs. If the system has been to enabling-new-function mode or new-function mode, CMON returns it to conversion* mode. Conversion* mode is similar to conversion mode, but the * indicates that at one time the system was in enabling-new-function mode or new-function mode. You can still access objects that were created in enabling-new-function mode or new-function mode. Data sharing groups cannot have any Db2 10 members. You cannot fall back to Db2 10 from conversion* mode or coexist with a Db2 10 system.

CONVERT
Starts enabling-new-function mode processing for the table space that is listed after the INPUT keyword.

INPUT table-space-name
Specifies the table space for which enabling-new-function mode processing should begin.

Before converting the catalog

Certain activities might be required before you run the CATMAINT utility, depending on your situation.

Before you run the CATENFM utility to convert the catalog, take image copies of all catalog and directory objects and save your entire subsystem.

To convert the catalog, you must run the DSNTIJEN job.

Data sets that CATENFM uses when converting the catalog

The CATENFM utility uses a number of data sets during its operation.

A CATENFM job allocates all of the data sets that it needs. CATENFM uses data sets only when the CONVERT option is specified.
The following table lists the data sets that CATENFM uses during conversion. The table lists the DD name that is used to identify the data set, a description of the data set, and an indication of whether it is required.

Table 6. Data sets that CATENFM uses during conversion

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSIN</td>
<td>Input data set that contains the utility control statement.</td>
<td>Yes</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>Output data set for messages.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Related concepts:
"Data sets that online utilities use" on page 16

Concurrency and compatibility for CATENFM

The CATENFM utility has certain concurrency and compatibility characteristics associated with it.

Certain catalog and directory objects are not available during some of the CATENFM phases. The objects that are unavailable vary based on the CATENFM option that you specify. The unavailability of these objects can cause other jobs to time out with message DSNT376I or DSNT501I. You cannot run CATENFM when the Db2 catalog or directory are in UT status.

Converting to new-function mode

The Db2 subsystem leaves conversion mode and enters enabling-new-function mode when you invoke CATENFM START by running the DSNTIJEN job.

About this task

When you migrate to Db2 11, the Db2 subsystem enters conversion mode. In conversion mode, the Db2 subsystem can coexist with other data sharing members that are at either Db2 10 or Db2 11 conversion mode.

Procedure

To convert to Db2 11 new-function mode:

Invoke CATENFM START by running the DSNTIJEN job.

The subsystem cannot begin enabling-new-function mode processing if any Db2 10 members are active in the data sharing group. All members, including members that are not converting to Db2 11 new-function mode, must be running Db2 11 when the subsystem enters enabling-new-function mode. Note that when a member starts enabling-new-function mode, the group enters enabling-new-function mode.

After enabling-new-function mode completes, the Db2 subsystem can enter Db2 11 new-function mode. All new Db2 11 functions are unavailable until the Db2 subsystem enters new-function mode.

The DSNTIJEN job runs CATENFM START, which causes the Db2 subsystem to enter enabling-new-function mode. Run CATENFM START only when you are ready to begin the enabling-new-function mode conversion process.
Termination or halt of CATENFM

You can terminate CATENFM by using the TERM UTILITY command.

You can stop the enabling-new-function mode processing by specifying CATENFM HALTENFM or by running job DSNTIJNH. Either action stops the enabling-new-function mode processing at the completion of the step that is currently executing.

CATENFM CONVERT cannot be restarted. If you attempt to restart CATENFM CONVERT, you receive message DSNU181I, which states that the utility cannot be restarted. You must terminate the job, and rerun job DSNTIJEN from the beginning to convert the catalog.

Related reference:
- TERM UTILITY (Db2) (Db2 Commands)

Related information:
- DSNU181I (Db2 Messages)
Chapter 10. CATMAINT

The CATMAINT online utility updates the catalog; run this utility during migration to Db2 11 or when instructed by IBM Support.

You can tailor and run the DSNTIJTC job to run the CATMAINT utility. For more information, see [Migration step 18: Tailor Db2 11 catalog: DSNTIJTC (Db2 Installation and Migration)](http://www.ibm.com) or [Installation step 15: Tailor the Db2 catalog: DSNTIJTC (Db2 Installation and Migration)](http://www.ibm.com).

**Output**

Output for CATMAINT UPDATE is the updated catalog.

**Authorization required**

The required authorization for CATMAINT is installation SYSADM.

**Execution phases of CATMAINT**

The CATMAINT utility operates in the following phases:

1. UTILINIT performs initialization.
2. UTILTERM performs cleanup.

If the catalog contains plans or packages that were bound with DBPROTOCOL(PRIVATE), the CATMAINT utility executes successfully; however, plans and packages that were bound with DBPROTOCOL(PRIVATE) and access remote locations cannot execute in DB2 10 and later. To enable the plans or packages to execute, convert them to use the DRDA protocol by rebinding them.

**Before running CATMAINT**

During migration, the work file database is used for CATMAINT sorting. If you are migrating from a previous version, calculate the size of the work file database.

**Data sets for CATMAINT**

Include DD statements for all data sets that your job uses. The following table lists the data sets that CATMAINT uses. The table lists the DD name that is used to identify the data set, a description of the data set, and an indication of whether it is required.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSIN</td>
<td>An input data set that contains the utility control statement</td>
<td>Yes</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>An output data set for messages</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Concurrency and compatibility for CATMAINT

Many catalog and directory indexes are not available while CATMAINT is running. The unavailability of these indexes can cause other jobs to time out with messages DSNT318I, DSNT376I or DSNT501I.

CATMAINT syntax and options

Use the ISPF/PDF edit function to create a control statement and to save it in a sequential or partitioned data set. When you create the JCL for running the job, use the SYSIN DD statement to specify the name of the data set that contains the utility control statement.

```
CATMAINT UPDATE
```

UPDATE
Indicates that you want to update the catalog.

```
switch-spec:
```

```
SCHEMA SWITCH(schema-name,new-schema-name)
```

```
OWNER FROM (owner-name) TO ROLE
```

```
VCAT SWITCH(catalog-name,new-catalog-name)
```

**SCHEMA SWITCH**(schema-name,new-schema-name)
Changes the owner, creator, and schema of database objects. The authorization IDs of the creator or owner for plans and packages that use the objects are not changed.

* schema-name is a string that identifies the existing owner, creator, or schema to be changed. It will be ignored if it does not identify any owner, creator, or schema names.

* schema-name cannot identify a schema or qualifier of an object on which any of the following objects depend:
  * Triggers
  * Views
  * SQL functions
  * Materialized query tables
  * Native SQL procedures
• Expression-based indexes
• Column masks
• Row permissions

`schema-name` cannot be referenced in a check condition in any check constraints. Ownership of objects will not be changed if the owner is a role.

`new-schema-name` specifies the new name for the owner, creator, or schema. The name cannot be a schema that qualifies existing objects.

**OWNER FROM(owner-name) TO ROLE**
Changes the ownership of objects from a user to a role. A trusted context must have been created for INSTALL SYSADM before CATMAINT UPDATE OWNER can run. The authorization IDs of the creator or owner for plans and packages that use the objects are not changed.

`owner-name` specifies the current owner of the object. You can specify multiple owners.

**VCAT SWITCH(catalog-name,new-catalog-name)**
Changes the catalog name that is used by storage groups, user indexes, and table spaces.

`catalog-name`
Identifies the integrated catalog facility catalog that is currently used by user-managed data sets for indexes, table spaces, and storage groups.

`new-catalog-name`
Specifies the new user-managed data sets for indexes, table spaces, and storage groups.

The data sets are linear VSAM data sets cataloged in the integrated catalog facility catalog that `catalog-name` identifies. For more information about `catalog-name` values, see [Naming conventions (Db2 SQL)](#).

More than one Db2 subsystem can share the integrated catalog facility catalogs with the current server. To avoid the chance of those subsystems attempting to assign the same name to different data sets, specify a `catalog-name` value that is not used by the other Db2 subsystems.

To specify any non-alphanumeric characters, enclose each name in single quotes.

```
utilx-spec
```

**UTILX**
Updates the catalog and OBD to the state specified (BASIC or EXTENDED).

• If the table space is already in the requested state no change is made, and a successful message and return code is given.
If the table space is different from the requested state and the table space is not empty, it fails with a DSNU777I message.

If the table space is different from the requested state and the table space is empty, the table space is reset to the new state.

**BASIC**

Initializes SYSUTILX and its indexes to basic 6-byte RBA format. Ensure that all utilities are completed or terminated before you change the format or the command will fail with a DSNU777I message. This option is available in all modes except conversion mode.

**EXTENDED**

Initializes SYSUTILX and its indexes to extended 10-byte RBA format. Ensure that all utilities are completed or terminated before you change the format or the command will fail with a DSNU777I message. This option is available in all modes except conversion mode.

**RESET**

Reinitializes the DSNDB01.SYSUTILX directory table space. Because DSNDB01.SYSUTILX contains information about active and outstanding utilities, the process of reinitializing this table space involves determining which objects have a utility in progress and resolving any pending states to make the object available for access. This option is available in conversion mode.

Reinitialize the DSNDB01.SYSUTILX directory table space in any of the following situations:

- You cannot successfully run the DISPLAY UTILITY or TERMINATE UTILITY commands.

After you run this statement, DSNDB01.SYSUTILX is reset to an empty state, and the previous contents are lost. If there were active or stopped utilities at that time, their tracking information is lost and the subsystem might experience unpredictable results. It is important that all utilities that can be terminated are terminated before you run UPDATE UTILX RESET.

**Termination or restart of CATMAINT**

You can terminate CATMAINT by using the TERM UTILITY command, but the termination might leave some indexes in REBUILD-pending status.

CATMAINT cannot be restarted. If you attempt to restart CATMAINT, you receive message DSNU181I, which states that the utility cannot be restarted. You must terminate the job with the TERM UTILITY command, and rerun CATMAINT from the beginning.

**Related tasks:**

- Migration step 18: Tailor Db2 11 catalog: DSNTIJTC (Db2 Installation and Migration)
- Installation step 15: Tailor the Db2 catalog: DSNTIJTC (Db2 Installation and Migration)
- “Reinitializing DSNDB01.SYSUTILX” on page 493

**Updating the catalog for installation or migration to Db2 11**

When you install or migrate to a new release of Db2, you must update the catalog for the prior release to the new version.
Procedure

Run job DSNTIJTC. For details, see Installation step 15: Tailor the Db2 catalog: DSNTIJTC (Db2 Installation and Migration) or Migration step 18: Tailor Db2 catalog: DSNTIJTC (Db2 Installation and Migration). Job DSNTIJTC runs the CATMAINT UPDATE to update the catalog. Db2 displays migration status message DSNU777I at several points during CATMAINT execution. If an abend occurs during migration processing, message DSNU776I or DSNU778I can give you information about the problem.

Related information:

- DSNU776I (Db2 Messages)
- DSNU777I (Db2 Messages)
- DSNU778I (Db2 Messages)

Renaming the owner, creator, and schema of database objects

You can rename the owner, creator, and schema of database objects.

Procedure

To rename the owner, creator, and schema of database objects:

Run the CATMAINT utility with the SCHEMA SWITCH options. This process updates every owner, creator or schema name in the catalog and directory that matches the schema_name value. All grants that were made by or received by the original owner are changed to the new owner. You can change multiple names by repeating the SWITCH keyword, but you can not specify the same name more than once. The names cannot be longer than 8 bytes in EBCDIC representation. 'SYSIBM' is not allowed as a schema_name or new_schema_name. OWNER FROM and SCHEMA SWITCH are mutually exclusive. You cannot specify both clauses in the same CATMAINT UPDATE statement.

Changing the ownership of objects from an authorization ID to a role

If a Db2 role is the owner of an object, all users that are associated with that role have the same owner privileges. You can change the owner of an object from an authorization ID to a role by using the CATMAINT utility.

Before you begin

You must be running under a trusted context with a role.

Procedure

To change the ownership of objects from an authorization ID to a role:

Run CATMAINT with the OWNER FROM owner_name TO ROLE clause. The current role will become the owner. All privileges that are held on the object will be transferred from the original owner to the role with the exception of plans and packages. The original user can be the grantor or grantee. After the utility completes, the original owner does not have any privileges to the object. You can change multiple object owners by specifying multiple owner_name values, but you cannot specify the same name more than once. If the owner_name value does not own any objects, that value is ignored. SYSIBM is not allowed as an
Ownership of roles is changed like other objects. However, if the associated trusted context role is owned by \textit{owner\_name}, the ownership of the role will not be changed, because a role cannot be owned by itself.

\textbf{Related reference:}

- Chapter 10, “CATMAINT,” on page 65

\textbf{Related information:}

- [Authorization IDs and roles (Managing Security)](#)

### Changing the catalog name used by storage groups or index spaces and table spaces

You can use the CATMAINT online utility to change the catalog name that is used by data sets for storage groups, by index spaces, and table spaces.

#### About this task

The data sets are linear VSAM data sets cataloged in the integrated catalog facility catalog that \texttt{catalog\_name} identifies. For more information about \texttt{catalog\_name} values, see [Naming conventions (Db2 SQL)](#).

More than one Db2 subsystem can share the integrated catalog facility catalogs with the current server. To avoid the chance of those subsystems attempting to assign the same name to different data sets, specify a \texttt{catalog\_name} value that is not used by the other Db2 subsystems.

#### Procedure

To change the catalog name that is used by data sets for storage groups, index spaces, and table spaces:

Run the CATMAINT utility and specify the VCAT SWITCH keywords. The VCAT SWITCH keyword is similar to the USING VCAT clause of the ALTER TABLESPACE statement for changing the catalog name. You must move the data for the affected indexes or table spaces to the data set on the new catalog in a separate step. You can change multiple names by repeating the SWITCH keyword, but you cannot specify the same name more than once. The VCAT SWITCH option has no effect on the system indexes and table spaces in DSNDB06 or DSNDB01 because the catalog name is maintained in the parameter.

\textbf{Related reference:}

- [CREATE TABLESPACE (Db2 SQL)](#)
- [Data set naming conventions (Db2 Administration Guide)](#)

### Identifying invalidated packages after the owner, creator, or schema name of an object is renamed

When the schema name of an object is changed, any packages that are dependent on the object are invalidated. Automatic rebind occurs when the invalidated package is executed.
About this task

Rebind might not be successful if the object is referenced in the application explicitly with the original schema name. In this case, you need to modify the application. The following queries identify the packages that will be invalidated:

```sql
SELECT DISTINCT COLLID, NAME
  FROM SYSIBM.SYSPACKDEP, SYSIBM.SYSPACKAGE
  WHERE BQUALIFIER IN (schema_name1, schema_name2...)
  ORDER BY COLLID, NAME;
```
Chapter 11. CHECK DATA

The CHECK DATA online utility checks table spaces for violations of referential and table check constraints. This utility also checks for consistency between a base table space and the corresponding LOB or XML table spaces.

CHECK DATA also verifies data consistency in hash access tables and checks the integrity of XML documents and their related node ID indexes. CHECK DATA does not check LOB table spaces or informational referential constraints.

After a conditional restart or a point-in-time recovery, run CHECK DATA on all table spaces where parent and dependent tables or base and auxiliary tables might not be synchronized.

Restriction: Do not run CHECK DATA on data that is encrypted through built-in functions. Because CHECK DATA does not decrypt that data, the utility might produce unpredictable results.

Output

CHECK DATA SHRLEVEL REFERENCE optionally copies and deletes rows that violate referential or table check constraints. The utility copies each row that violates one or more constraints to an exception table. If a row violates two or more constraints, the utility copies the row only once. CHECK DATA SHRLEVEL REFERENCE resets CHECK-pending status if it finds no errors or if all rows that contain violations were copied to exception tables and deleted.

CHECK DATA SHRLEVEL CHANGE operates on shadow copies of the table space and generates REPAIR statements to correct any violations.

CHECK DATA does not set CHECK-pending status on a table space, even if it finds violations of constraints in the table space. If you want CHECK DATA to set CHECK-pending status on a table space when violations or constraints are detected, set the CHECK_SETCHKP subsystem parameter to YES.

Authorization required

To execute this utility, you must use a privilege set that includes one of the following authorities:
• STATS privilege for the database
• DBADM, DBCTRL, or DBMAINT authority for the database. If the object on which the utility operates is in an implicitly created database, DBADM authority on the implicitly created database or DSNDB04 is required.
• DATAACCESS authority
• SYSCTRL or SYSADM authority

An ID with installation SYSOPR authority can execute CHECK DATA on only SYSUTILX in database DSNDB01.

If you specify SHRLEVEL CHANGE, the user ID that invokes COPY with the CONCURRENT option must provide the necessary authority to execute the DFSMSdss COPY command. DFSMSdss creates a shadow data set with the
authority of the utility batch address space. The submitter should have an RACF ALTER authority, or its equivalent, for the shadow data set.

If you specify the DELETE option, the privilege set must include the DELETE privilege on the tables that are being checked. If you specify the FOR EXCEPTION option, the privilege set must include the INSERT privilege on any exception table that is used. If you specify AUXERROR INVALIDATE, LOBERROR INVALIDATE, or XMLERROR INVALIDATE, the privilege set must include the UPDATE privilege on the base tables.

Execution phases of CHECK DATA

<table>
<thead>
<tr>
<th>Phase</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTILINIT</td>
<td>Initializes utility processing.</td>
</tr>
<tr>
<td>CHECKXML</td>
<td>Checks XML structures for all XML table spaces that are specified by INCLUDE XML TABLESPACES.</td>
</tr>
<tr>
<td>SCANTAB</td>
<td>Extracts foreign keys. The utility uses an index if the index contains the same columns or a superset of the columns in the foreign key. Otherwise, the utility scans the table.</td>
</tr>
<tr>
<td>SORT</td>
<td>Sorts foreign keys if they are not extracted from the foreign key index.</td>
</tr>
<tr>
<td>CHECKDAT</td>
<td>Looks in primary indexes for foreign key parents, checks XML schema validations, checks XML structure, and issues messages to report detected errors.</td>
</tr>
<tr>
<td>REPORTCK</td>
<td>Copies error rows into exception tables and deletes them from the source table if DELETE YES is specified.</td>
</tr>
<tr>
<td>UTILTERM</td>
<td>Performs cleanup.</td>
</tr>
</tbody>
</table>

Syntax and options of the CHECK DATA control statement

The CHECK DATA utility control statement, with its multiple options, defines the function that the utility job performs.

You can create a control statement with the ISPF/PDF edit function. After you create it, save it in a sequential or partitioned data set. When you create the JCL for running the job, use the SYSIN DD statement to specify the name of the data set that contains the utility control statement.
Syntax diagram

Notes:
1. If you specify AUXERROR and LOBERROR or XMLERROR, the options for the keywords (REPORT and INVALIDATE) must match.
xml-column-spec:

```
     ┌───┐
     │ TABLE  │
     │ schema-name ─ table-name ── XMLCOLUMN ─ column-name │
     └───┘
```

table-space-spec:

```
     ┌───┐
     │ TABLESPACE  │
     │ database-name ─ table-space-name │
     └───┘
```

drain-spec:

```
     ┌───┐
     │ DRAIN_WAIT ─ integer ─ RETRY ─ integer ─ RETRY_DELAY ─ integer │
     └───┘
```

**Option descriptions**

**DATA**
Indicates that you want the utility to check referential and table check constraints. CHECK DATA does not check informational referential constraints.

**TABLESPACE database-name.table-space-name**
Specifies the table space to which the data belongs. You can specify base table spaces or, if TABLESPACE is specified as a part of the INCLUDE XML TABLESPACES option, XML table spaces. TABLESPACE cannot be used to specify LOB table spaces.

database-name is the name of the database and is optional. The default value is DSNDB04.

table-space-name is the name of the table space.

**PART integer**
Identifies which partition to check for constraint violations.

integer is the physical partition number. It must be in the range from 1 to the number of partitions that are defined for the table space. The maximum is 4096.
**INCLUDE XML TABLESPACES**

Indicates that CHECK DATA is to perform consistency checks on the specified XML table spaces and related node ID indexes.

By default, the utility checks only the XML table spaces and their related node ID indexes. If an XML type modifier exists for an XML column and *xml-spec* is specified, XML documents can also be checked against the stored XML schemas. Specify XMLSCHEMA on the *xml-spec* option to enable the check against stored XML schemas.

The consistency checks enabled by INCLUDE XML TABLESPACE are performed in addition to the existing checks specified by the SCOPE keyword. XML indexes that are associated with the XML table spaces that are checked are not verified. Run the CHECK INDEX utility separately on those indexes.

The following checks are performed:

- The XML table space is checked to ensure that all rows of each XML document are present in the XML table space and that the XML document is structurally intact.

- All entries in the node ID index are checked against the rows in the XML table space. Each index entry must have a corresponding row in the XML table space, and vice versa. This functionality is equivalent to running the CHECK INDEX utility on the node ID index.

- All values in the document ID column are checked against the node ID index. Each document ID value must have matching entries in the node ID index. Each node ID index value must also have a document ID value.

- If XMLSCHEMA is specified, CHECK DATA validates documents that are stored in that column. When a document is validated, the base table row is updated with the validated document that is returned when SHRLEVEL REFERENCE and XMLError INVALIDATE or AUXERROR INVALIDATE are specified.

**ALL**

Checks all XML table spaces that are related to the base table spaces that are identified by the *table-space-spec*. Specifying ALL is equivalent to explicitly specifying all the XML column identifiers.

**xml-spec**

Checks only those XML table spaces and related node ID indexes that are identified by either the XML column of a table or by the explicit table space name.

Each XML column has a single XML table space that is associated with it. Therefore, an XML table space can be identified either by the XML column of the base table or by the explicit table space name.

If an XML column identifier is used, the utility finds the name of the XML table space in the Db2 catalog or the database directory.

**table-space-spec**

Identifies an XML table space to check. The XML table space specification must identify an XML table space that has a corresponding column in a base table. The base table must reside in the table space that is identified by the *table-space-spec* option of the main CHECK DATA control statement.

**xml-column-spec**

Identifies an XML table space to check by the XML column of the XML table space in a base table. An XML column identifier consists of the
fully qualified table name and the name of the XML column. An XML column identifier must reference a table in any one of the base table spaces that are to be checked.

**XMLSCHEMA**

Specifies that if the XML columns have an XML type modifier, the CHECK DATA utility checks the XML documents against the stored XML schema.

**CLONE**

Indicates that CHECK DATA is to check the clone table in the specified table space. Because clone tables cannot have referential constraints, the utility checks only constraints for inconsistencies between the clone table data and the corresponding LOB data. If you do not specify CLONE, CHECK DATA operates against only the base table.

**SHRLEVEL**

Indicates the type of access that is to be allowed for the index, table space, or partition that is to be checked during CHECK DATA processing.

**REFERENCE**

Specifies that applications can read from but cannot write to the index, table space, or partition that is to be checked.

The CHECK DATA utility can write changes to the table space, index space, or partition during processing.

**Restriction:** You cannot run CHECK DATA with the SHRLEVEL REFERENCE option on a table space that contains an archive-enabled table or system-period temporal table when one of the following options is also specified:

- DELETE YES
- LOBERROR INVALIDATE
- AUXERROR INVALIDATE
- XMLERROR INVALIDATE

**CHANGE**

Specifies that applications can read from and write to the index, table space, or partition that is to be checked.

The CHECK DATA utility operates on shadow copies only and does not change the table space, index space, or partition during processing. REPAIR statements are generated for any changes to be made and are written to the data set that is indicated in the PUNCHDDN option. CHECK DATA does not generate REPAIR statements for inconsistencies that it finds in compressed rows if you specify SHRLEVEL CHANGE and one of the following options:

- AUXERROR INVALIDATE
- LOBERROR INVALIDATE
- XMLERROR INVALIDATE

If you specify SHRLEVEL CHANGE, Db2 performs the following actions:

- Drains all writers and forces the buffers to disk for the specified object and all of its indexes
- Invokes DFSMSdss to copy the specified object and all of its indexes to shadow data sets
- Enables read/write access for the specified object and all of its indexes...
• Runs CHECK INDEX on the shadow data sets

By default, DFSMSdss uses FlashCopy to copy Db2 objects to shadow data sets, if FlashCopy is available. If DFSMSdss cannot use FlashCopy, DFSMSdss uses a slower method. As a result, creating copies of objects might take a long time, and the time during which the data and indexes have read-only access might increase. You can set the CHECK_FASTREPLICATION subsystem parameter to REQUIRED to force the CHECK utility to use only FlashCopy. If FlashCopy is not available, the CHECK utility fails.

Related information:

````Subsystem parameters for refining DFSMSdss COPY operation with utilities” on page 40
````

**DRAIN_WAIT**

Specifies the number of seconds that CHECK DATA is to wait when it drains the table space or index. The specified time is the aggregate time for objects that are to be checked. This value overrides the values that are specified by the IRLMRWT and UTIMOUT subsystem parameters.

`integer` can be any integer from 0 to 1800. If you do not specify DRAIN_WAIT or specify a value of 0, CHECK DATA uses the value of the lock timeout subsystem parameter IRLMRWT.

**RETRY** `integer`

Specifies the maximum number of retries that CHECK DATA is to attempt.

`integer` can be any integer from 0 to 255. If you do not specify RETRY, CHECK DATA uses the value of the utility multiplier system parameter UTIMOUT.

Specifying RETRY can increase processing costs and result in multiple or extended periods during which the specified index, table space, or partition is in read-only access.

**RETRY_DELAY** `integer`

Specifies the minimum duration, in seconds, between retries. `integer` can be any integer from 1 to 1800.

If you do not specify RETRY_DELAY, CHECK DATA uses the smaller of the following two values:

- DRAIN_WAIT value \( \times \) RETRY value
- DRAIN_WAIT value \( \times 10 \)

**SCOPE**

Limits the scope of the rows in the table space that are to be checked.

**PENDING**

Indicates that the only rows to be checked are those rows that are in table spaces or partitions that are in CHECK-pending (CHKP) status. For each or these rows, CHECK DATA checks all referential and check constraints and LOB and XML columns.

When CHECK DATA checks XML columns, the utility verifies the relationship between the node ID index and the values in the XML indicator column in the base table space. If the INCLUDE XML TABLESPACES option is specified, CHECK DATA also validates the schema for all specified XML table spaces that satisfy both of the following conditions:

- Are in CHKP status
• Reference a table in any of the base table spaces to be checked

If you specify SCOPE PENDING for a table space that is not in CHKP status, CHECK DATA does not check the table space. The utility does not issue an error message.

**AUXONLY**
Indicates that only the LOB and XML columns are to be checked for table spaces that have tables with LOB columns or XML columns. CHECK DATA does not check referential and check constraints.

When CHECK DATA checks XML columns, the utility verifies only the relationship between the node ID index and the values in the XML indicator column in the base table space.

**ALL**
Indicates that all dependent tables in the specified table spaces are to be checked. CHECK DATA checks all referential and check constraints and LOB and XML columns.

If the INCLUDE XML TABLESPACES option is specified, the associated XML table space and node ID index are checked for structural defects and inconsistencies.

**REFOONLY**
Indicates the same behavior as the ALL option, except that the LOB and XML columns are not checked. CHECK DATA checks all referential and check constraints.

**XMLSCHEMAONLY**
Indicates that only the XML schema is to be validated for the XML objects that are specified by the INCLUDE XML TABLESPACE option. CHECK DATA does not check XML and LOB column integrity or referential and check constraints.

**AUXERROR**
Specifies the action that CHECK DATA is to perform when it finds a LOB or XML column check error.

**REPORT**
A LOB or XML column check error is reported with a warning message. The base table space is set to the auxiliary CHECK-pending (ACHKP) status.

**Note:** CHECK DATA sets the base table space to ACHKP status if SHRLEVEL REFERENCE is specified. If SHRLEVEL CHANGE is specified, CHECK DATA does not change the status of the base table space.

**INVALIDATE**
A LOB or XML column check error is reported with a warning message. The base table LOB or XML column is set to an invalid status. A LOB or XML column with invalid status that is now correct is set valid. This action is also reported with a message. The base table space is set to the auxiliary warning (AUXW) status if any LOB column remains in invalid status.

If SHRLEVEL REFERENCE is specified, CHECK DATA sets the base table of a LOB or XML column to an invalid status and the base table space to AUXW status. If SHRLEVEL CHANGE is specified, CHECK DATA does not change the status of the base table space or a LOB or XML column.

If SHRLEVEL REFERENCE and INCLUDE XML TABLESPACES are specified, CHECK DATA deletes corrupted XML documents and the
associated node ID index entries. If the node ID index is not consistent with the content in the XML table, CHECK DATA corrects the node ID index.

**Restrictions:** You cannot run CHECK DATA SHRLEVEL REFERENCE with AUXERROR INVALIDDATE on the following objects:
- A table or a history table that is defined with data versioning
- A table space that contains an archive-enabled table

Before you use CHECK DATA to check a LOB or XML column, take the following actions:
1. Run CHECK LOB to ensure the validity of the LOB table space.
2. Run REBUILD INDEX or CHECK INDEX on the index on the auxiliary table to ensure its validity.
3. Run REBUILD INDEX or CHECK INDEX on the NODE ID index on the XML table space to ensure its validity.

**LOBERROR**
Specifies the action that CHECK DATA is to perform when it finds a LOB column check error. Do not specify LOBERROR if AUXERROR is specified. If both are specified, the keywords must match. LOBERROR is ignored for SCOPE XMLONLY since LOB checking is not being performed.

**REPORT**
A LOB column check error is reported with a warning message. The base table space is set to the auxiliary CHECK-pending (ACHKP) status.

If AUXERROR is not specified, the default value is **REPORT**.

**INVALIDATE**
A LOB column check error is reported with a warning message. The base table LOB column is set to an invalid status. A LOB column with invalid status that is now correct is set valid. The base table space is set to the auxiliary warning (AUXW) status if any LOB column remains in invalid status.

**Restrictions:** You cannot run CHECK DATA with LOBERROR INVALIDDATE on the following objects:
- A table or a history table that is defined with data versioning
- A table space that contains an archive-enabled table if SHRLEVEL REFERENCE is also specified

**XMLERROR**
Specifies the action that CHECK DATA is to perform when it finds an XML column check error. Do not specify XMLERROR if AUXERROR is specified. If both are specified, the keywords must match. XMLERROR is ignored for SCOPE XMLONLY since LOB checking is not being performed.

**REPORT**
An XML column check error is reported with a warning message. The base table space is set to the auxiliary CHECK-pending (ACHKP) status.

If AUXERROR is not specified, the default value is **REPORT**.

**Note:** CHECK DATA sets the base table space to ACHKP status if SHRLEVEL REFERENCE is specified. If SHRLEVEL CHANGE is specified, CHECK DATA does not change the status of the base table space.
INVALIDATE
An XML column check error is reported with a warning message. The base
table XML column is set to an invalid status. An XML column with invalid
status that is now correct is set valid. The base table space is set to the
auxiliary warning (AUXW) status if any LOB column remains in invalid
status.
CHECK DATA sets the base table of a LOB or XML column to an invalid
status and the base table space to AUXW only if SHRLEVEL REFERENCE
is specified. If SHRLEVEL CHANGE is specified, CHECK DATA does not
change the status of the base table space or a LOB or XML column.
If SHRLEVEL REFERENCE and INCLUDE XML TABLESPACES are
specified, CHECK DATA deletes corrupted XML documents and the
associated node ID index entries. If the node ID index is not consistent
with the content in the XML table, CHECK DATA corrects the node ID
index.

Restrictions: You cannot run CHECK DATA SHRLEVEL REFERENCE with
XMLERROR INVALIDATE on the following objects:
- A table or a history table that is defined with data versioning
- A table space that contains an archive-enabled table

FOR EXCEPTION
Indicates that any row that is in violation of referential or table check
constraints is to be copied to an exception table. Although this keyword does
not apply to the checking of LOB or XML columns, rows with LOB or XML
columns are moved to the exception tables. If you specify AUXONLY for LOB
and XML checking only, the FOR EXCEPTION option is ignored.
If any row violates more than one constraint, that row is included only once in
the exception table. CHECK DATA includes checking for XML schema
violations and XML structure checking.
This option is ignored when SHRLEVEL CHANGE is specified.
If you run CHECK DATA on a base table with XML columns, the
EXCEPTIONS keyword has an effect only if the INCLUDE XML
TABLESPACES option is also specified.
IN table-name1
Specifies the table (in the table space that is specified on the TABLESPACE
keyword) from which rows are to be copied.
table-name1 is the name of the table.

USE table-name2
Specifies the exception table into which error rows are to be copied.
table-name2 is the name of the exception table and must be a base table; it
cannot be a view, synonym, or alias.

For both table-name1 and table-name2, enclose the table name in quotation
marks if the name contains a blank or a special character. (A special character is
any character other than a letter or a digit.)

DELETE
Indicates whether rows that are in violation of referential or table check
constraints are to be deleted from the table space.
You can specify DELETE only if you specify the FOR EXCEPTION clause.
**NO** Indicates that error rows are to remain in the table space. Primary errors in dependent tables are copied to exception tables.

If DELETE NO and SHRLEVEL REFERENCE are specified, and constraint violations are detected, CHECK DATA places the table space in the CHECK-pending status.

**YES** Indicates that error rows are to be deleted from the table space.

When you specify FOR EXCEPTION, deleted rows from both dependent and descendant tables are placed into exception tables.

If you specify SHRLEVEL REFERENCE, error rows are deleted from the table space. If you specify SHRLEVEL CHANGE, CHECK DATA generates REPAIR LOCATE DELETE statements that you can run to delete the rows. These statements are written to the PUNCHDDN data set.

**Important:** Check any generated REPAIR statements after you run CHECK DATA SHRLEVEL CHANGE on tables that have data versioning activated or on history tables. Historic information could be deleted.

If you delete rows from a table space that is not logged, the table space is placed in informational COPY-pending (ICOPY) status.

**Restrictions:** You cannot run CHECK DATA with DELETE YES on the following objects:

- A table or a history table that is defined with data versioning
- A table space that contains an archive-enabled table if SHRLEVEL REFERENCE is also specified

**LOG**

Specifies the logging action that is to be taken when records are deleted.

**YES** Logs all records that are deleted during the REPORTCK PHASE.

If the table space has the NOT LOGGED attribute, LOG YES is ignored.

**NO**

Does not log any records that are deleted during the REPORTCK phase. Other types of log records are still written. If any rows are deleted, CHECK DATA places the table space in COPY-pending status and any indexes with the COPY YES attribute in informational COPY-pending status. If rows are deleted from a table space that is not logged, the table space is marked informational COPY-pending.

**Attention:** Use the LOG NO option with caution because its use limits your ability to recover data by using the log. For example, suppose that you issue a CHECK DATA DELETE YES LOG NO statement at particular log RBA. You can recover data that exists on the log before that point in time or after the point on the log at which the utility execution completes.

**EXCEPTIONS** integer

Specifies the maximum number of exceptions, which are reported by messages only. CHECK DATA terminates in the CHECKDATA phase when it reaches the specified number of exceptions; if termination occurs, the error rows are not written to the EXCEPTION table.
Only records that contain primary referential integrity errors or table check constraint violations are applied toward the exception limit. The number of records that contain secondary errors is not limited.

\textit{integer} is the maximum number of exceptions. The default value is 0, which indicates no limit on the number of exceptions.

This keyword does not apply to LOB table spaces or base table spaces that contain XML columns.

**ERRDDN** \texttt{ddname}

Specifies a DD statement for an error processing data set.

\texttt{ddname} is either a DD name or a TEMPLATE name specification from a previous TEMPLATE control statement. If utility processing detects that the specified name is both a DD name in the current job step and a TEMPLATE name, the utility uses the DD name. The default value is \texttt{SYSERR}.

**WORKDDN** \texttt{(ddname1, ddname2)}

Specifies the DD statements for the temporary work file for sort input and the temporary work file for sort output. A temporary work file for sort input and output is required.

You can use the WORKDDN keyword to specify either a DD name or a TEMPLATE name specification from a previous TEMPLATE control statement. If utility processing detects that the specified name is both a DD name in the current job step and a TEMPLATE name, WORKDDN uses the DD name.

\texttt{ddname1} is the DD name of the temporary work file for sort input. The default is \texttt{SYSUT1}.

\texttt{ddname2} is the DD name of the temporary work file for sort output. The default is \texttt{SORTOUT}.

**PUNCHDDN** \texttt{ddname}

Specifies the DD statement for a data set that is to receive the REPAIR utility control statements that CHECK DATA SHRLEVEL CHANGE generates.

\texttt{ddname} is the DD name.

The default value is \texttt{SYSPUNCH}.

The PUNCHDDN keyword specifies either a DD name or a TEMPLATE name specification from a previous TEMPLATE control statement. If utility processing detects that the specified name is both a name in the current job step and a TEMPLATE name, the utility uses the DD name.

**SORTDEVT** \texttt{device-type}

Specifies the device type for temporary data sets that are to be dynamically allocated by a sort program. You can specify any disk device type that is acceptable to the DYNALLOC parameter of the SORT or OPTION control statement for the sort program. Tape devices are not supported by the sort program.

Do not use a TEMPLATE specification to dynamically allocate sort work data sets. The presence of the SORTDEVT keyword controls dynamic allocation of these data sets.

\texttt{device-type} is the device type. If you omit SORTDEVT and a sort is required, you must provide the DD statements that the sort program requires for the temporary data sets.
SORTNUM integer
Specifies the number of temporary data sets that are to be dynamically allocated by the sort program.

integer is the number of temporary data sets that can range from 2 to 255.

If you omit SORTDEVT, SORTNUM is ignored. If you use SORTDEVT and omit SORTNUM, no value is passed to the sort program; the sort program uses its own default.

You need at least two sort work data sets for each sort. The SORTNUM value applies to each sort invocation in the utility.

Important: The SORTNUM keyword is ignored if the UTSORTAL subsystem parameter is set to YES and the IGNSORTN subsystem parameter is set to YES.

Related reference:
Chapter 34, “TEMPLATE,” on page 793

Before running CHECK DATA

Certain activities might be required before you run the CHECK DATA utility, depending on your situation.

For a table with no LOB columns
Before running CHECK DATA, you should run CHECK INDEX on primary key indexes and foreign key indexes to ensure that the indexes that CHECK DATA uses are valid. This action is especially important before using CHECK DATA with the DELETE YES or PART options.

For a table with LOB columns
If you plan to run CHECK DATA on a base table space that contains at least one LOB column, complete the following steps prior to running CHECK DATA:
1. Run CHECK LOB on the LOB table space.
2. Run CHECK INDEX on the index on the auxiliary table to ensure the validity of the LOB table space and the index on the auxiliary table.
3. Run CHECK INDEX on the indexes on the base table space.

The relationship between a base table with a LOB column and the LOB table space is shown in the following figure. The LOB column in the base table points to the auxiliary index on the LOB table space, as illustrated in the figure.
If the LOB table space is in either the CHECK-pending or RECOVER-pending status, or if the index on the auxiliary table is in REBUILD-pending status, CHECK DATA issues an error message and fails.

Complete all LOB column definitions. You must complete all LOB column definitions for a base table before running CHECK DATA. A LOB column definition is not complete until the LOB table space, auxiliary table, and index on the auxiliary table have been created. If any LOB column definition is not complete, CHECK DATA fails and issues error message DSNU075E.

**For an XML table space**

Before running CHECK DATA, run CHECK INDEX on the node ID index of each XML column. If you need to determine the XML objects, query the SYSXMLRELS catalog table.

**Data sets that CHECK DATA uses**

The CHECK DATA utility uses a number of data sets during its operation.

The following table lists the data sets that CHECK DATA uses. The table lists the DD name that is used to identify the data set, a description of the data set, and an indication of whether it is required. Include statements in your JCL for each required data set and any optional data sets that you want to use.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSIN</td>
<td>An input data set that contains the utility control statement.</td>
<td>Yes</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>An output data set for messages.</td>
<td>Yes</td>
</tr>
<tr>
<td>Work data sets</td>
<td>Two temporary data sets for sort input and sort output. Specify the DD names by using the WORKDDN option of the utility control statement. The default ddname for sort input is SYSUT1. The default ddname for sort output is SORTOUT.</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Table 8. Data sets that CHECK DATA uses (continued)

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error data set</td>
<td>An output data set that collects information about violations that are</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>encountered during the CHECKDAT phase for referential constraints or the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SCANTAB phase for check constraints. Specify the DD name by using</td>
<td></td>
</tr>
<tr>
<td></td>
<td>the ERRDDN parameter of the utility control statement. The default ddbname</td>
<td></td>
</tr>
<tr>
<td></td>
<td>is SYSERR.</td>
<td></td>
</tr>
<tr>
<td>UTPRINT</td>
<td>A data set that contains messages from the sort program (usually, SYSOUT or</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>DUMMY).</td>
<td></td>
</tr>
</tbody>
</table>

The following objects are named in the utility control statement and do not require DD statements in the JCL:

**Table space**
Object that is to be checked. (If you want to check only one partition of a table space, use the PART option in the control statement.)

**Exception table**
Table that stores rows that violate any referential constraints. For each table in a table space that is checked, specify the name of an exception table in the utility control statement. Any row that violates a referential constraint is copied to the exception table.

Sort work data sets cannot span volumes. Smaller volumes require more sort work data sets to sort the same amount of data; therefore, large volume sizes can reduce the number of needed sort work data sets. It is recommended that at least 1.2 times the amount of data to be sorted be provided in sort work data sets on disk.

Tape devices are not supported for sort work data sets.

**Related concepts:**
“Data sets that online utilities use” on page 16

**Related reference:**
Db2 Sort for z/OS

**Related information:**
DFSORT Application Programming Guide

**Defining work data sets**
Three sequential data sets are required during execution of CHECK DATA. Two work data sets and one error data set are described by DD statements in the WORKDDN and ERRDDN options.

**Procedure**
To define work data sets:
1. Find the approximate size, in bytes, of the WORKDDN data set:
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>If a table space has a LOB column</td>
<td>Count a total of 70 bytes for the LOB column and multiply the sum by the number of keys and LOB columns that are checked.</td>
</tr>
<tr>
<td>If a table space does not have a LOB column</td>
<td>Add 18 to the length of the longest foreign key. For nonpadded indexes, the length of the longest foreign key is the maximum possible length of the key with all varying-length columns in the key padded to their maximum length, plus 2 bytes for each varying-length column.</td>
</tr>
</tbody>
</table>

2. Create the ERRDDN data set so that it is large enough to accommodate one error entry (length=60 bytes) per violation that CHECK DATA detects.

**Shadow data sets for CHECK DATA**

When you execute the CHECK DATA utility with the SHRLEVEL CHANGE option, the utility uses shadow data sets.

If a table space, partition, or index resides in Db2-managed data sets and shadow data sets do not already exist when you execute CHECK DATA, Db2 creates the shadow data sets. At the end of CHECK DATA processing, the Db2-managed shadow data sets are deleted.

For user-managed data sets, DFSMSdss can create, or scratch and re-create, the required shadow data sets as needed. When the CHECK DATA utility completes the processing of user-managed data sets, the shadow data sets are not automatically scratched.

If you do not want the shadow data sets to be allocated in the same storage class as the production data sets, set the UTIL_TEMP_STORCLAS system parameter to specify the storage class for the shadow data sets.

**Shadow data set names**

Each shadow data set must have the following name:

`catname.DSNDBx.dbname.psname.y000z.Lnnn`

In the preceding name, the variables have the following meanings:

**variable**

**meaning**

catname

- The VSAM catalog name or alias

x

- C or D

dbname

- Database name

psname

- Table space name or index name

y

- I or J
**Lnnn** Partition identifier. Use one of the following values:
- A001 through A999 for partitions 1 through 999
- B000 through B999 for partitions 1000 through 1999
- C000 through C999 for partitions 2000 through 2999
- D000 through D999 for partitions 3000 through 3999
- E000 through E996 for partitions 4000 through 4096

To determine the names of existing data sets, execute one of the following queries against the SYSTABLEPART or SYSINDEXPART catalog tables:

```sql
SELECT DBNAME, TSNAME, IPREFIX
FROM SYSIBM.SYSTABLEPART
WHERE DBNAME = 'dbname'
AND TSNAME = 'psname';
```

```sql
SELECT DBNAME, IXNAME, IPREFIX
FROM SYSIBM.SYSINDEXES X, SYSIBM.SYSINDEXPART Y
WHERE X.NAME = Y.IXNAME
AND X.CREATOR = Y.IXCREATOR
AND X.DBNAME = 'dbname'
AND X.INDEXSPACE = 'psname';
```

For a partitioned table space, Db2 returns rows from which you select the row for the partitions that you want to check.

**Defining shadow data sets**

For a partitioned table space, Db2 returns rows from which you select the row for the partitions that you want to check.

Consider the following actions when you preallocate the data sets:
- Allocate the shadow data sets according to the rules for user-managed data sets.
- Define the shadow data sets as LINEAR.
- Use SHAREOPTIONS(3,3).
- Define the shadow data sets as EA-enabled if the original table space or index space is EA-enabled.
- Allocate the shadow data sets on the volumes that are defined in the storage group for the original table space or index space.

If you specify a secondary space quantity, Db2 does not use it. Instead, Db2 uses the SECQTY value for the table space or index space.

**Recommendation:** Use the MODEL option, which causes the new shadow data set to be created like the original data set. This method is shown in the following example:

```sql
DEFINE CLUSTER +
(NAME('catname.DSNDBC.dbname.psname.x0001.L001') +
MODEL('catname.DSNDBC.dbname.psname.y0001.L001')) +
```
Creating shadow data sets for indexes

When you preallocate shadow data sets for indexes, create the data sets as follows:
- Create shadow data sets for the partition of the table space and the corresponding partition in each partitioning index and data-partitioned secondary index.
- Create a shadow data set for logical partitions of nonpartitioned secondary indexes.

Use the same naming scheme for these index data sets as you use for other data sets that are associated with the base index, except use J0001 instead of I0001.

Estimating the size of shadow data sets

If you have not changed the value of FREEPAGE or PCTFREE, the amount of required space for a shadow data set is comparable to the amount of required space for the original data set.

Concurrency and compatibility for CHECK DATA

The CHECK DATA utility has certain concurrency and compatibility characteristics associated with it.

Db2 treats individual data and index partitions as distinct target objects. Utilities that operate on different partitions of the same table space or index space are compatible.

Claims and drains

The following table shows which claim classes CHECK DATA claims and drains and any restrictive status that the utility sets on the target object. The legend for these claim classes is located at the bottom of the table.

<table>
<thead>
<tr>
<th>Target objects</th>
<th>CHECK DATA PART DELETE NO</th>
<th>CHECK DATA PART DELETE YES</th>
<th>CHECK DATA PART DELETE NO</th>
<th>CHECK DATA PART DELETE YES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table space or partition</td>
<td>DW/UTRO</td>
<td>DA/UTUT</td>
<td>DW/UTRO</td>
<td>DA/UTUT</td>
</tr>
<tr>
<td>Partitioning index or index partition</td>
<td>DW/UTRO</td>
<td>DA/UTUT</td>
<td>DW/UTRO</td>
<td>DA/UTUT</td>
</tr>
<tr>
<td>Secondary index</td>
<td>DW/UTRO</td>
<td>DA/UTUT</td>
<td>none</td>
<td>DR</td>
</tr>
<tr>
<td>Logical partition of index</td>
<td>none</td>
<td>none</td>
<td>DW/UTRO</td>
<td>DA/UTUT</td>
</tr>
<tr>
<td>Primary index</td>
<td>DW/UTRO</td>
<td>DW/UTRO</td>
<td>DW/UTRO</td>
<td>DW/UTRO</td>
</tr>
<tr>
<td>RI dependent and descendent table spaces and indexes</td>
<td>none</td>
<td>DA/UTUT</td>
<td>none</td>
<td>DA/UTUT</td>
</tr>
</tbody>
</table>
Table 9. Claim classes of CHECK DATA operations (continued)

<table>
<thead>
<tr>
<th>Target objects</th>
<th>CHECK DATA DELETE</th>
<th>CHECK DATA PART DELETE</th>
<th>CHECK DATA DELETE</th>
<th>CHECK DATA PART DELETE</th>
</tr>
</thead>
<tbody>
<tr>
<td>RI exception table spaces and indexes (FOR EXCEPTION only)</td>
<td>DA/UTUT</td>
<td>DA/UTUT</td>
<td>DA/UTUT</td>
<td>DA/UTUT</td>
</tr>
</tbody>
</table>

Legend:
- DA: Drain all claim classes, no concurrent SQL access
- DR: Drain the repeatable read class, no concurrent access for SQL repeatable readers
- DW: Drain the write claim class, concurrent access for SQL readers
- UTUT: Utility restrictive state, exclusive control
- UTRO: Utility restrictive state, read-only access allowed
- none: Object not affected by this utility
- RI: Referential Integrity

The following table shows claim classes on a LOB table space and an index on the auxiliary table.

Table 10. Claim classes of CHECK DATA operations on a LOB table space and index on the auxiliary table

<table>
<thead>
<tr>
<th>Target objects</th>
<th>CHECK DATA DELETE</th>
<th>CHECK DATA DELETE</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOB table space</td>
<td>DW/UTRO</td>
<td>DA/UTUT</td>
</tr>
<tr>
<td>Index on the auxiliary table</td>
<td>DW/UTRO</td>
<td>DA/UTUT</td>
</tr>
</tbody>
</table>

Legend:
- DW: Drain the write claim class, concurrent access for SQL readers
- DA: Drain all claim classes, no concurrent SQL access
- UTRO: Utility restrictive state, read-only access allowed
- UTUT: Utility restrictive state, exclusive control

The following table shows claim classes of XML objects.

Table 11. Claim classes of XML objects

<table>
<thead>
<tr>
<th>Target objects</th>
<th>CHECK DATA DELETE</th>
<th>CHECK DATA DELETE</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML table space</td>
<td>DW/UTRO</td>
<td>DA/UTUT</td>
</tr>
<tr>
<td>document ID and node ID indexes</td>
<td>DW/UTRO</td>
<td>DA/UTUT</td>
</tr>
<tr>
<td>XML index</td>
<td>DW/UTRO</td>
<td>DA/UTUT</td>
</tr>
</tbody>
</table>

Legend:
- DW: Drain the write claim class, concurrent access for SQL readers
- DA: Drain all claim classes, no concurrent SQL access
- UTRO: Utility restrictive state, read-only access allowed
- UTUT: Utility restrictive state, exclusive control

When you specify CHECK DATA AUXERROR INVALIDATE, a drain-all is performed on the base table space, and the base table space is set UTUT.
Compatibility

The following utilities are compatible with CHECK DATA and can run concurrently on the same target object:
- DIAGNOSE
- MERGECOPY
- MODIFY
- REPORT
- STOSPACE
- UNLOAD (when CHECK DATA DELETE NO)

SQL operations and other online utilities are incompatible.

To run on DSNDB01.SYSUTILX, CHECK DATA must be the only utility in the job step and the only utility that is running in the Db2 subsystem.

The index on the auxiliary table for each LOB column inherits the same compatibility and concurrency attributes of a primary index.

Exception tables for the CHECK DATA utility

An exception table is a user-created table that duplicates the definition of a dependent table. The CHECK DATA utility checks the number of columns in the dependent table. The CHECK DATA utility also copies the deleted rows from the dependent table to the exception table.

The following table describes the contents of an exception table. This table lists the columns, a description of the column content, whether or not the column is required, the data type and length of the column value, and whether or not the column has the NULL attribute.

<table>
<thead>
<tr>
<th>Table 12. Contents of exception tables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column</td>
</tr>
<tr>
<td>--------------------</td>
</tr>
<tr>
<td>1 to n</td>
</tr>
<tr>
<td>n+1</td>
</tr>
<tr>
<td>n+2</td>
</tr>
<tr>
<td>≥ n+2</td>
</tr>
</tbody>
</table>
### Table 12. Contents of exception tables (continued)

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
<th>Required?</th>
<th>Data type and length</th>
<th>NULL attribute</th>
<th>Scale of data type</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>You can use CHAR(5) for any type of table space, but you must use it for table spaces that are defined with the LARGE or DSSIZE options.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If you delete rows by using the CHECK DATA utility with SCOPE ALL, you must create exception tables for all tables that are named in the table spaces and for all their descendents. All descendents of any row are deleted.

When creating or using exception tables, be aware of the following guidelines:

- The exception tables should not have any unique indexes or referential or table check constraints that might cause errors when CHECK DATA inserts rows into them.
- You can create a new exception table before you run CHECK DATA, or you can use an existing exception table. The exception table can contain rows from multiple invocations of CHECK DATA.
- If column \( n+2 \) is of type TIMESTAMP, CHECK DATA records the starting time. Otherwise, it does not use column \( n+2 \).
- You must have DELETE authorization on the dependent table that is being checked.
- You must have INSERT authorization on the exception table.
- Column names in the exception table can have any name.
- Any change to the structure of the dependent table (such as a dropped column) is not automatically recorded in the exception table. You must make that change in the exception table.

**Related reference:**

[CREATE TABLE (Db2 SQL)]

### Exception processing for tables with auxiliary columns

CHECK DATA writes constraint violations to exception tables. The exception table for the base table must have a similar auxiliary column and an auxiliary table space for each auxiliary column.

If an exception is found, Db2 moves the base table row with its auxiliary column to the exception table. If you specify DELETE YES, Db2 deletes the base table row and the auxiliary column.

An auxiliary table cannot be an exception table. A LOB column check error is not included in the exception count. A row with only a LOB column check error does not participate in exception processing.

### Specifying the scope of CHECK DATA

Running CHECK DATA with SCOPE PENDING is normally sufficient. Db2 records which data rows must be checked to ensure the referential integrity of the table space.
About this task

You can find inconsistencies in the XML table space, the node ID index, or in the relationship between the document ID column and the node ID index by running the CHECK DATA utility.

Running CHECK DATA with SCOPE ALL or SCOPE AUXONLY and specifying INCLUDE XML TABLESPACES enables the XML structure checking of the specified XML table spaces and consistency checking of the XML columns in the base table and their associated node ID indexes. Specifying XMLSCHEMAONLY with INCLUDE XML TABLESPACES limits the CHECK DATA scope to only XML schema validation for the XML columns.

Procedure

To specify the scope of CHECK DATA:

Use one of the following approaches:

- If the scope information is in doubt, run the utility with the SCOPE ALL option. The scope information is recorded in the Db2 catalog. The scope information can become indoubt whenever you start the target table space with ACCESS(FORCE), or when the catalog is recovered to a point in time.
- If you want to check only the tables with LOB columns, specify the AUXONLY option. If you want to check all dependent tables in the specified table spaces except tables with LOB columns, specify the REFONLY option.

How violations are identified

CHECK DATA issues a message for every row that contains a referential constraint violation or table check constraint violation.

The violation is identified by:

- The RID of the row
- The name of the table that contains the row
- The name of the constraint that is being violated

The following figure shows an example of messages that CHECK DATA issues.
Detection and correction of constraint violations

You can avoid problems by running CHECK DATA with DELETE NO to detect violations before you attempt to correct the errors.

If required, use DELETE YES after you analyze the output and understand the errors.

You can automatically delete rows that violate referential or table check constraints by specifying CHECK DATA with DELETE YES. However, you should be aware of the following possible problems:

- The violation might be created by a non-referential integrity error. For example, the indexes on a table might be inconsistent with the data in a table.
- Deleting a row might cause a cascade of secondary deletes in dependent tables. The cascade of deletes might be especially inconvenient within referential integrity cycles.
- The error might be in the parent table.

CHECK DATA uses the primary key index and all indexes that exactly match a foreign key. Therefore, before running CHECK DATA, ensure that the indexes are consistent with the data by using the CHECK INDEX utility.

CHECK DATA XML error detection

Run CHECK DATA with the INCLUDE XML TABLESPACES option to verify the consistency of the XML table space and the node ID index.

The following checks are performed:

```
DSNU0501  DSNUGUTC - CHECK DATA TABLESPACE DBJM1203.TLJM1203 TABLESPACE DBJM1203.TBJM1204 FOR EXCEPTION IN TLJM1203.TBJM1204 USE ADMF001.EXCPT3 IN TPJM1204.TBJM1204 USE ADMF001.EXCPT4 DELETE YES
DSNU7271 = DSNUKINP - TABLESPACE 'DBJM1203.TLJM1203' IS NOT CHECK PENDING
DSNU7301  DSNUKOST - CHECKING TABLE TPJM1204.TBJM1204
DSNU0421  DSNUGSRD - SORT PHASE STATISTICS - NUMBER OF RECORDS=4 ELAPSED TIME=00:00:00
DSN07331 DSNUKERK - ROW (RID=X'0000000020B') HAS NO PARENT FOR TPJM1204.TBJM1204.TABFK
DSN07331 DSNUKERK - ROW (RID=X'0010000201') HAS NO PARENT FOR TPJM1204.TBJM1204.TABFK
DSN07331 DSNUKERK - ROW (RID=X'0020000020B') HAS NO PARENT FOR TPJM1204.TBJM1204.TABFK
DSN07331 DSNUKERK - ROW (RID=X'00300000201') HAS NO PARENT FOR TPJM1204.TBJM1204.TABFK
DSNU7391  DSNUKDAT - CHECK TABLE TPJM1204.TBJM1204 COMPLETE, ELAPSED TIME=00:00:00
DSNU7411 = DSNUKRDY - 4 ROWS DELETED FROM TABLE TPJM1204.TBJM1204
DSNU5681 = DSNUGSRX - INDEX TPJM1204.IPJM1204.IS IS IN INFORMATIONAL COPY PENDING
DSNU5681 = DSNUGSRX - INDEX TPJM1204.IXJM1204.IS IS IN INFORMATIONAL COPY PENDING
DSNU7491 DSNUK001 - CHECK DATA COMPLETE, ELAPSED TIME=00:00:02
DSNU0101  DSNUGBC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=4
```

Figure 7. Example of messages that CHECK DATA issues
• Verify that all rows that comprise an XML document exist in the XML table space and that all nodes in that XML document are structurally intact.
• Verify that the node ID index is consistent with the content in the XML table space. No index entries must exist without an associated XML document and each XML document in the XML table space must have corresponding entries in the node ID index.
• Verify that the references from the base table space ID column contains only entries that can be found in the node ID index in the XML table space. Also verify that the node ID index does not contain any entries for which no matching value in the document ID column in the base table space can be found.

Any inconsistencies found are reported as errors. All remaining parts of corrupted XML documents will be deleted from the XML table space. All the associated node ID index entries for the affected XML document will be deleted and the XML column in the base table will be set to an invalid status.

When running with SHRLEVEL CHANGE, CHECK DATA operates on shadow copies of the table spaces to be checked, corresponding REPAIR statements are generated. These generated statements must be executed by the REPAIR utility to perform the mandatory actions which CHECK DATA has identified.

Two REPAIR statements are generated.
• One statement deletes the corrupted XML document and its associated node ID index entries.
• The other REPAIR statement sets the XML column in the base table to an invalid status.

Correcting XML data after running CHECK DATA

After you run the CHECK DATA utility, you might need to correct XML data.

Procedure

To correct XML data after running CHECK DATA:

Based on the CHECK DATA output, perform one of the following actions:

<table>
<thead>
<tr>
<th>Problem</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem with corrupted XML data</td>
<td>REPAIR statements are generated to delete each corrupted XML document from the XML table space and its associated node ID index entry.</td>
</tr>
<tr>
<td>Problem with document ID index</td>
<td>Run generated REPAIR LOCATE TABLESPACE control statements.</td>
</tr>
<tr>
<td>Problem with node ID index</td>
<td>Run generated REPAIR LOCATE TABLESPACE control statements.</td>
</tr>
<tr>
<td>Problem with integrity of XML column in the base table and the node ID index</td>
<td></td>
</tr>
</tbody>
</table>
Resetting CHECK-pending status

If a table space has a status of CHECK-pending, you can remove the CHECK-pending status by correcting the error and running a utility job. You can either rerun the CHECK DATA utility with SHRLEVEL REFERENCE specified or you can run the REPAIR utility.

Procedure

To remove CHECK-pending status by running the CHECK DATA utility, use the following approaches:

- Use the DELETE NO option if no tables contain rows that violate referential or table check constraints. If referential or table check constraint violations are found, the table space or partition is placed in CHECK-pending status.
- Use the DELETE YES option to remove all rows that violate referential or table check constraints.

Related reference:
“CHECK-pending status” on page 1039

LOB column errors

If you run CHECK DATA on a base table space that contains at least one LOB column, you might receive an error on the LOB column.

If you specify CHECK DATA AUXERROR REPORT, AUXERROR INVALIDATE, LOBERRO REPORT, or LOBERRO INVALIDATE and a LOB column check error is detected, Db2 issues a message that identifies the table, row, column, and type of error. Any additional actions depend on the option that you specify for the AUXERROR or LOBERRO parameter:

When you specify the AUXERROR REPORT or LOBERRO REPORT option
Db2 sets the base table space to the auxiliary CHECK-pending (ACHKP) status. If CHECK DATA encounters only invalid LOB columns and no other LOB column errors, the base table space is set to the auxiliary warning (AUXW) status.

When you specify the AUXERROR INVALIDATE or LOBERRO INVALIDATE option
Db2 sets the base table LOB columns that are in error to an invalid status. Db2 resets the invalid status of LOB columns that have been corrected. If any invalid LOB columns remain in the base table, Db2 sets the base table space to auxiliary warning (AUXW) status. You can use SQL to update a LOB column that is in the AUXW status; however, any other attempt to access the column results in a -904 SQL return code.

If you run CHECK DATA AUXERROR REPORT or INVALIDATE on a base table space that contains at least one LOB column, the following errors might be reported:

Orphan LOBs

An orphan LOB column is a LOB that is found in the LOB table space but that is not referenced by the base table space. If an orphan error is the only type of error reported by CHECK DATA, the base table is considered correct.

An orphan can result from the following situations:
A table space with LOB or XML columns can be recovered to a point in time. In this case, RECOVER TABLESPACE sets the auxiliary CHECK-pending (ACHKP) status on the table space. You can remove the auxiliary CHECK-pending status if Db2 does not find any inconsistencies.

About this task

Use one of the following actions to reset auxiliary CHECK-pending status:

Procedure

To reset auxiliary CHECK-pending status:

Take one of the following actions:
• Use the SCOPE ALL option to check all dependent tables in the specified table space. The checks include referential integrity constraints, table check constraints, and the existence of LOB and XML columns.

• Use the SCOPE PENDING option to check table spaces or partitions with CHKP status. The checks include referential integrity constraints, table check constraints, and the existence of LOB and XML columns.

• Use the SCOPE AUXONLY option to check for LOB and XML columns.

Results

If you specified the AUXERROR INVALIDATE, LOBERROE INVALIDATE or XMLERROR INVALIDATE option and Db2 finds inconsistencies, it places the table space in AUXW status.

Related reference:

Appendix B, “Advisory or restrictive states,” on page 1037

Termination and restart of CHECK DATA

You can terminate and restart the CHECK DATA utility.

When you terminate CHECK DATA, table spaces remain in the same CHECK-pending status as they were at the time the utility was terminated. The CHECKDAT phase places the table space in the CHECK-pending status when CHECK DATA detects an error; at the end of the phase, CHECK DATA resets the CHECK-pending status if it detects no errors. The REPORTCK phase resets the CHECK-pending status if you specify the DELETE YES option.

You can restart a CHECK DATA utility job, but it starts from the beginning again.

Related tasks:

“Restarting an online utility” on page 44

“Terminating an online utility” on page 43

Sample CHECK DATA control statements

Use sample control statements as models for developing your own CHECK DATA control statements.

Example 1: Copying violations into exception tables

The control statement specifies that the CHECK DATA utility is to check for and delete any rows that violate referential and table check constraints in table spaces DSN8D11A.DSN8S11D and DSN8D11A.DSN8S11E. CHECK DATA copies any rows that violate these constraints into the exception tables that are specified in the FOR EXCEPTION clause. For example, CHECK DATA is to copy the violations in table DSN8810.DEPT into table DSN8810.EDEPT.

```
//STEP1 EXEC DSNUPROC,UID='IUIQU1UQ.CHK1', // UTPROC='', // SYSTEM='DSN'
//SYSUT1 DD DSN='IUIQU1UQ.CHK3.STEP1.SYSUT1',DISP=(MOD,DELETE,CATLG), // UNIT='SYSDA',SPACE=(8000,(20,20),,,ROUND)
//SYSERR DD DSN='IUIQU1UQ.CHK3.SYSERR',DISP=(MOD,DELETE,CATLG), // UNIT='SYSDA',SPACE=(6000,(20,20),,,ROUND)
//SORTOUT DD DSN='IUIQU1UQ.CHK3.STEP1.SORTOUT',DISP=(MOD,DELETE,CATLG), // UNIT='SYSDA',SPACE=(6000,(20,20),,,ROUND)
//SYSIN DD *
CHECK DATA TABLESPACE DSN8D11A.DSN8S11D
```
Example 2: Creating an exception table for the project activity table

You can create an exception table for the project activity table by using the following SQL statements:

```
EXEC SQL
CREATE TABLE EPROJACT
LIKE DSN8B10.PROJACT
    IN DATABASE DSN8D11A
ENDEXEC

EXEC SQL
ALTER TABLE EPROJACT
    ADD RID CHAR(4)
ENDEXEC

EXEC SQL
ALTER TABLE EPROJACT
    ADD TIME TIMESTAMP NOT NULL WITH DEFAULT
ENDEXEC
```

The first statement requires the SELECT privilege on table DSN8B10.PROJACT and the privileges that are usually required to create a table.

Table EPROJACT has the same structure as table DSN8B10.PROJACT, but it can have two extra columns. The columns in EPROJACT are:

- Its first five columns mimic the columns of the project activity table; they have exactly the same names and descriptions. Although the column names are the same, they do not need to be. However, the rest of the column attributes for the initial columns must be same as those of the table that is being checked.

- The next column, which is added by ALTER TABLE, is optional; CHECK DATA uses it as an identifier. The name “RID” is an arbitrary choice; if the table already has a column with that name, use a different name. The column description, CHAR(4), is required.

- The final timestamp column is also optional. If you define the timestamp column, a row identifier (RID) column must precede this column. You might define a permanent exception table for each table that is subject to referential or table check constraints. You can define it once and use it to hold invalid rows that CHECK DATA detects. The TIME column allows you to identify rows that were added by the most recent run of the utility.

Eventually, you correct the data in the exception tables, perhaps with an SQL UPDATE statement, and transfer the corrections to the original tables by using statements that are similar to those in the following example:
Example 3: Running CHECK DATA on a table space with LOBs

Assume that table space DBIQUQ01.TPIQU01 contains LOB columns. In the following control statement, the SCOPE ALL option indicates that CHECK DATA is to check all rows in all dependent tables in table space DBIQUQ01.TPIQU01 for the following violations:
- Violations of referential constraints
- Violations of table check constraints
- Inconsistencies between the base table space and the corresponding LOB table space.

The AUXERROR INVALIDATE option indicates that if the CHECK DATA utility finds a LOB column error in this table space, it is to perform the following actions:
- Issues a warning message
- Sets the base table LOB column to an invalid status
- Sets the base table to auxiliary warning (AUXW) status

Example 4: Specifying the maximum number of exceptions

The control statement specifies that the CHECK DATA utility is to check all rows in partition number 254 in table space DBNC0216.TPNC0216. The EXCEPTIONS 1 option indicates that the utility is to terminate when it finds one exception. Any exceptions are to be reported by messages only.
Example 5: Running CHECK DATA SHRLEVEL CHANGE

The control statement specifies that the CHECK DATA utility is to specifies that applications can read from and write to the table space that is to be checked.
CHECK DATA TABLESPACE DBNI0101.TSNI010P SHRLEVEL CHANGE

Example 6: Checking several table spaces

To check several table spaces, you can specify more than one table space in a CHECK DATA control statement. This technique is useful for checking a complete set of referentially related table spaces. The following example shows a CHECK DATA control statement that lists more than one table space.

CHECK DATA TABLESPACE DBJM1203.TLJM1203 TABLESPACE DBJM1203.TPJM1204 FOR EXCEPTION IN TLJM1203.TBJM1203 USE ADMF001.EXCPT3 IN TPJM1204.TMBJM1204 USE ADMF001.EXCPT4 DELETE YES

Example 7: Checking XML columns

The control statement specifies how to include consistency checking of XML columns in a base table with the associated node ID indexes. Specify XMLSCHEMAONLY with INCLUDE XML TABLESPACES to limit the CHECK DATA scope to only XML schema validation for the XML columns.

CHECK DATA TABLESPACE DBNI0101.TSNI010P INCLUDE XML TABLESPACES SCOPE XMLSCHEMAONLY AUXONLY

Example 8: Running CHECK DATA on a clone table

The control statement specifies that the CHECK DATA utility is to check the clone table in the specified table space.
CHECK DATA TABLESPACE DBNI0101.TSNI010P CLONE SCOPE ALL ERRODN SYSSERR
Chapter 12. CHECK INDEX

The CHECK INDEX online utility tests whether indexes are consistent with the data that they index, and it issues warning messages when it finds an inconsistency.

Run the CHECK INDEX utility after a conditional restart or a point-in-time recovery on all table spaces whose indexes might not be consistent with the data.

Also run CHECK INDEX before running CHECK DATA, especially if you specify DELETE YES. Running CHECK INDEX before CHECK DATA ensures that the indexes that CHECK DATA uses are valid. When checking an auxiliary table index, CHECK INDEX verifies that each LOB is represented by an index entry, and that an index entry exists for every LOB.

**Important:** Inaccurate statistics for tables, table spaces, or indexes can result in a sort failure during CHECK INDEX.

**Running CHECK INDEX when the index has a VARBINARY column**

If you run CHECK INDEX against the index with the following characteristics, CHECK INDEX fails:

- The index was created on a VARBINARY column or a column with a distinct type that is based on a VARBINARY data type.
- The index column has the DESC attribute.

To fix the problem, alter the column data type to BINARY, and then rebuild the index.

**Output**

CHECK INDEX generates several messages that show whether the indexes are consistent with the data.

For unique indexes, any two null values are treated as equal values, unless the index was created with the UNIQUE WHERE NOT NULL clause. In that case, if the key is a single column, it can contain any number of null values, and CHECK INDEX does not issue an error message.

CHECK INDEX issues an error message if it finds two or more null values and the unique index was not created with the UNIQUE WHERE NOT NULL clause.

**Authorization required**

To execute this utility, you must use a privilege set that includes one of the following authorities:

- STATS privilege for the database
- DBADM, DBCTRL, or DBMAINT authority for the database. If the object on which the utility operates is in an implicitly created database, DBADM authority on the implicitly created database or DSNDB04 is required.
- System DBADM authority
• SYSCTRL or SYSADM authority

An ID with installation SYSOPR authority can also execute CHECK INDEX, but only on a table space in the DSNDB01 or DSNDB06 databases.

If you are using SHRLEVEL CHANGE, the user ID that invokes COPY with the CONCURRENT option must provide the necessary authority to execute the DFSMSdss ADRDSSU command. DFSMSdss will create a shadow data set with the authority of the utility batch address space. The submitter should have an RACF ALTER authority, or its equivalent, for the shadow data set.

**Execution phases of CHECK INDEX**

<table>
<thead>
<tr>
<th>Phase</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTILINIT</td>
<td>Performs initialization</td>
</tr>
<tr>
<td>UNLOAD</td>
<td>Unloads data keys</td>
</tr>
<tr>
<td>SORTCHK</td>
<td>Sorts unloaded data keys and scans the index to validate data keys.</td>
</tr>
<tr>
<td>UTILTERM</td>
<td>Performs cleanup</td>
</tr>
</tbody>
</table>

**Syntax and options of the CHECK INDEX control statement**

The CHECK INDEX utility control statement, with its multiple options, defines the function that the utility job performs.

You can create a control statement with the ISPF/PDF edit function. After creating it, save it in a sequential or partitioned data set. When you create the JCL for running the job, use the SYSIN DD statement to specify the name of the data set that contains the utility control statement.
Syntax diagram

Option descriptions

**INDEX**
Indicates that you are checking for index consistency.

**LIST listdef-name**
Specifies the name of a previously defined LISTDEF list name. The list must contain only index spaces. Do not specify the name of an index or of a table space. Db2 groups indexes by their related table space and executes CHECK INDEX once per table space. CHECK INDEX allows one LIST keyword for each control statement in CHECK INDEX. This utility processes clone data only if the CLONE keyword is specified. The use of CLONED YES on the LISTDEF statement is not sufficient.

The partitions or partition ranges can be specified in a list.

*(index-name, ...)*
Specifies the indexes that are to be checked. All indexes must belong to tables in the same table space. If you omit this option, you must use the (ALL) TABLESPACE option. Then, CHECK INDEX checks all indexes on all tables in the table space that you specify.

*index-name* is the name of an index, in the form *creator-id.name*. If you omit the qualifier *creator-id.*, the user identifier for the utility job is used. If you use a list of names, separate items in the list by commas. Parentheses are required around a name or list of names. Enclose the index name in quotation marks if the name contains a blank.

**PART integer**
Identifies a physical partition of a partitioned index or a logical partition of a nonpartitioned index that is to be checked for consistency. If you specify an index on a nonpartitioned table space, an error occurs.

*integer* is the physical partition number. It must be in the range from 1 to the number of partitions that are defined for the table space. The maximum is 4096.
If the PART keyword is not specified, CHECK INDEX tests the entire target index for consistency.

(ALL)
Specifies that all indexes in the specified table space that are referenced by the table space are to be checked.

TABLESPACE database-name.table-space-name
Specifies the table space from which all indexes are to be checked. If an explicit list of index names is not specified, all indexes on all tables in the specified table space are checked.

Do not specify TABLESPACE with an explicit list of index names.

database-name is the name of the database that the table space belongs to. The default value is DSNDB04.

table-space-name is the name of the table space from which all indexes are checked.

CLONE
Indicates that CHECK INDEX is to check only the specified indexes that are on clone tables. This utility processes clone data only if the CLONE keyword is specified. The use of CLONED YES on the LISTDEF statement is not sufficient.

SHRLEVEL
Indicates the type of access that is to be allowed for the index, table space, or partition that is to be checked during CHECK INDEX processing.

REFERENCE
Specifies that applications can read from but cannot write to the index, table space, or partition that is to be checked.

If you specify SHRLEVEL REFERENCE or use this value as the default, Db2 unloads the index entries, sorts the index entries, and scans the data to validate the index entries.

CHANGE
Specifies that applications can read from and write to the index, table space, or partition that is to be checked.

If you specify SHRLEVEL CHANGE, Db2 performs the following actions:
- Drains all writers and forces the buffers to disk for the specified object and all of its indexes
- Invokes DFSMSdss to copy the specified object and all of its indexes to shadow data sets
- Enables read/write access for the specified object and all of its indexes
- Runs CHECK INDEX on the shadow data sets

By default, DFSMSdss uses FlashCopy to copy Db2 objects to shadow data sets, if FlashCopy is available. If DFSMSdss cannot use FlashCopy, DFSMSdss uses a slower method. As a result, creating copies of objects might take a long time, and the time during which the data and indexes have read-only access might increase. You can set the CHECK_FASTREPLICATION subsystem parameter to REQUIRED to force the CHECK utility to use only FlashCopy. If FlashCopy is not available, the CHECK utility fails.

Related information:

“Subsystem parameters for refining DFSMSdss COPY operation with utilities” on page 40
DRAIN_WAIT integer
Specifies the number of seconds that CHECK INDEX is to wait when draining the table space or index. The specified time is the aggregate time for objects that are to be checked. This value overrides the values that are specified by the IRLMRWT and UTIMOUT subsystem parameters.

integer can be any integer from 0 to 1800. If you do not specify DRAIN_WAIT or specify a value of 0, CHECK INDEX uses the value of the lock timeout subsystem parameter IRLMRWT.

RETRY integer
Specifies the maximum number of times that CHECK INDEX is to keep trying to drain the object.

integer can be any integer from 0 to 255. Specifying a value other than 0 can increase processing costs and result in multiple or extended periods during which the specified index, table space, or partition is in read-only access.

If you do not specify RETRY, CHECK INDEX uses the value of the utility multiplier subsystem parameter UTIMOUT.

RETRY_DELAY integer
Specifies the minimum duration, in seconds, between attempts to drain the object. integer can be any integer from 1 to 1800.

If you do not specify RETRY_DELAY, CHECK INDEX uses the smaller of the following two values:
- DRAIN_WAIT value × RETRY value
- DRAIN_WAIT value × 10

SORTDEV device-type
Specifies the device type for temporary data sets that are to be dynamically allocated by the external sort program. You can specify any disk device type that is acceptable to the DYNALLOC parameter of the SORT or OPTION control statement for the sort program.

A TEMPLATE specification does not dynamically allocate sort work data sets. The SORTDEV keyword controls dynamic allocation of these data sets.

device-type is the device type. If you omit SORTDEV and a sort is required, you must provide the DD statements that the sort program requires for the temporary data sets.

SORTNUM integer
Specifies the number of temporary data sets that are to be dynamically allocated by the sort program.

integer is the number of temporary data sets that can range from 2 to 255.

If you omit SORTDEV, SORTNUM is ignored. If you use SORTDEV and omit SORTNUM, no value is passed to the sort program; the sort program uses its own default.

You need at least two sort work data sets for each sort. The SORTNUM value applies to each sort invocation in the utility. For example, suppose CHECK INDEX is checking three indexes, no constraints exist to limit parallelism, and you specify SORTNUM 8. In this case, a total of 24 sort work data sets are allocated for a job.

Each sort work data set consumes both above-the-line and below-the-line virtual storage. Therefore, if you specify a value for SORTNUM that is too high, the utility might decrease the degree of parallelism due to virtual storage...
constraints. The degree of parallelism might possibly be decreased down to one, which means that no parallelism is used.

**Important:** CHECK INDEX ignores the SORTNUM keyword if subsystem parameter UTSORTAL is set to YES and IGSORTN is set to YES.

**PARALLEL num-subtasks**

Specifies the maximum number of subtasks that are to be started in parallel to check indexes for consistency.

The value of `num-subtasks` must be an integer between 0 and 32767, inclusive. If the specified value for `num-subtasks` is greater than 32767, the CHECK INDEX statement fails. If 0 or no value is specified for `num-subtasks`, the CHECK INDEX utility uses the optimal number of parallel subtasks. If the specified value for `num-subtasks` is greater than the calculated optimal number, the CHECK INDEX utility limits the number of parallel subtasks to the optimal number with applied constraints. CHECK INDEX typically allocates subtasks in groups of two or three. Therefore, the actual number of subtasks that are started might be less than the number that is specified by the PARALLEL option.

The specified number of subtasks for PARALLEL always overrides the value of the PARAMDEG_UTIL subsystem parameter. Therefore, PARALLEL can be smaller or larger than the value of PARAMDEG_UTIL.

**Related information:**

- [Indexes in parallel](index_in_parallel) on page 114
- MAX UTILS PARALLELISM field (PARAMDEG_UTIL subsystem parameter) (Db2 Installation and Migration)

**Related reference:**

- Chapter 18, “LISTDEF,” on page 215
- Db2 Sort for z/OS

**Data sets that CHECK INDEX uses**

The CHECK INDEX utility uses a number of data sets during its operation.

The following table lists the data sets that CHECK INDEX uses. The table lists the DD name that is used to identify the data set, a description of the data set, and an indication of whether it is required. Include statements in your JCL for each required data set and any optional data sets that you want to use.

**Table 13. Data sets that CHECK INDEX uses**

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSIN</td>
<td>An input data set that contains the utility control statement.</td>
<td>Yes</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>An output data set for messages.</td>
<td>Yes</td>
</tr>
<tr>
<td>UTPRINT</td>
<td>A data set that contains messages from the sort program (usually, SYSOUT or DUMMY).</td>
<td>Yes</td>
</tr>
</tbody>
</table>
The following object is named in the utility control statement and does not require a DD statement in the JCL:

**Index space**

Object that is to be checked. (If you want to check only one partition of an index, use the PART option in the control statement.)

Sort work data sets cannot span volumes. Smaller volumes require more sort work data sets to sort the same amount of data; therefore, large volume sizes can reduce the number of needed sort work data sets. When you allocate sort work data sets on disk, the recommended amount of space to allow provides at least 1.2 times the amount of data that is to be sorted.

Tape devices are not supported for sort work data sets.

**Related concepts:**

“Data sets that online utilities use” on page 16

**Related reference:**

[Db2 Sort for z/OS](https://www.ibm.com/products/db2-sort-for-zos)

**Related information:**


---

**Shadow data sets for CHECK INDEX**

When you execute the CHECK INDEX utility with the SHRLEVEL CHANGE option, the utility uses shadow data sets.

If a table space, partition, or index resides in Db2-managed data sets and shadow data sets do not already exist when you execute CHECK INDEX, Db2 creates the shadow data sets. At the end of CHECK INDEX processing, the Db2-managed shadow data sets are deleted.

For user-managed data sets, DFSMSdss can create or scratch and re-create the required shadow data sets as needed. When the CHECK INDEX utility completes the processing of user-managed data sets, the shadow data sets are not automatically scratched.

If you do not want the shadow data sets to be allocated in the same storage class as the production data sets, set the UTIL_TEMP_STORCLAS system parameter to specify the storage class for the shadow data sets.

**Shadow data set names**

Each shadow data set must have the following name:

`catname.DSNDBx.dbname.psname.y000z.Lnnn`

In the preceding name, the variables have the following meanings:

<table>
<thead>
<tr>
<th>variable</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>catname</code></td>
<td>The VSAM catalog name or alias</td>
</tr>
<tr>
<td><code>x</code></td>
<td>C or D</td>
</tr>
</tbody>
</table>

---

Chapter 12. CHECK INDEX  109
**dbname**
Database name

**psname**
Table space name or index name

<table>
<thead>
<tr>
<th>y</th>
<th>I or J</th>
</tr>
</thead>
<tbody>
<tr>
<td>z</td>
<td>1 or 2</td>
</tr>
</tbody>
</table>

**Lnnn**
Partition identifier. Use one of the following values:
- A001 through A999 for partitions 1 through 999
- B000 through B999 for partitions 1000 through 1999
- C000 through C999 for partitions 2000 through 2999
- D000 through D999 for partitions 3000 through 3999
- E000 through E996 for partitions 4000 through 4096

---

To determine the names of existing data sets, execute one of the following queries against the SYSTABLEPART or SYSINDEXPART catalog tables:

```sql
SELECT DBNAME, TSNAME, IPREFIX
FROM SYSIBM.SYSTABLEPART
WHERE DBNAME = 'dbname'
AND TSNAME = 'psname';

SELECT DBNAME, IXNAME, IPREFIX
FROM SYSIBM.SYSINDEXES X, SYSIBM.SYSINDEXPART Y
WHERE X.NAME = Y.IXNAME
AND X.CREATOR = Y.IXCREATOR
AND X.DBNAME = 'dbname'
AND X.INDEXSPACE = 'psname';
```

---

**Defining shadow data sets**

For a partitioned table space, Db2 returns rows from which you select the row for the partitions that you want to check.

Consider the following actions when you preallocate the data sets:
- Allocate the shadow data sets according to the rules for user-managed data sets.
- Define the shadow data sets as LINEAR.
- Use SHAREOPTIONS(3,3).
- Allocate base or clone objects
- Define the shadow data sets as EA-enabled if the original table space or index space is EA-enabled.
- Allocate the shadow data sets on the volumes that are defined in the storage group for the original table space or index space.

If you specify a secondary space quantity, Db2 does not use it. Instead, Db2 uses the SECQTY value for the table space or index space.

**Recommendation:** Use the MODEL option, which causes the new shadow data set to be created like the original data set. This method is shown in the following example:
CREATE CLUSTER +
(NAME('catname.DSNDBC.dbname.psname.x000z.L001') +
  MODEL('catname.DSNDBC.dbname.psname.y000z.L001')) +
DATA +
(NAME('catname.DSNDBD.dbname.psname.x000z.L001') +
  MODEL('catname.DSNDBD.dbname.psname.y000z.L001'))

Creating shadow data sets for indexes

When you preallocate shadow data sets for indexes, create the data sets as follows:

- Create shadow data sets for the partition of the table space and the corresponding partition in each partitioning index and data-partitioned secondary index.
- Create a shadow data set for logical partitions of nonpartitioned secondary indexes.

Use the same naming scheme for these index data sets as you use for other data sets that are associated with the base index, except use J0001 instead of I0001.

Estimating the size of shadow data sets

If you have not changed the value of FREEPAGE or PCTFREE, the amount of required space for a shadow data set is comparable to the amount of required space for the original data set.

Concurrency and compatibility for CHECK INDEX

The CHECK INDEX utility has certain concurrency and compatibility characteristics associated with it.

Db2 treats individual data and index partitions as distinct target objects. Utilities that operate on different partitions of the same table space or index space are compatible.

Claims and drains

The following table shows which claim classes CHECK INDEX claims and drains and any restrictive state that the utility sets on the target object.

<table>
<thead>
<tr>
<th>Target</th>
<th>CHECK INDEX SHRLEVEL REFERENCE</th>
<th>CHECK INDEX PART SHRLEVEL REFERENCE</th>
<th>CHECK INDEX SHRLEVEL CHANGE</th>
<th>CHECK INDEX PART SHRLEVEL CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table space or partition</td>
<td>DW/UTRO</td>
<td>DW/UTRO</td>
<td>DW/UTRW</td>
<td>DW/UTRW</td>
</tr>
<tr>
<td>Partitioning index or index partition</td>
<td>DW/UTRO</td>
<td>DW/UTRO</td>
<td>DW/UTRW</td>
<td>DW/UTRW</td>
</tr>
<tr>
<td>Secondary index¹</td>
<td>DW/UTRO</td>
<td>none</td>
<td>DW/UTRW</td>
<td>DW/UTRW</td>
</tr>
<tr>
<td>Data-partitioned secondary index or index partition²</td>
<td>DW/UTRO</td>
<td>DW/UTRO</td>
<td>DW/UTRW</td>
<td>DW/UTRW</td>
</tr>
<tr>
<td>Logical partition of an index</td>
<td>none</td>
<td>DW/UTRO</td>
<td>DW/UTRW</td>
<td>DW/UTRW</td>
</tr>
</tbody>
</table>

Chapter 12. CHECK INDEX  111
Table 14. Claim classes of CHECK INDEX operations (continued)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SHRLEVEL</td>
<td>REFERENCE</td>
<td>SHRLEVEL</td>
<td>REFERENCE</td>
<td>SHRLEVEL</td>
</tr>
</tbody>
</table>

Legend:
- DW: Drain the write claim class, concurrent access for SQL readers
- UTR: Utility restrictive state, read only-access allowed
- UTRW: Utility restrictive state, read and write access allowed
- none: Object not affected by this utility

Note:
1. Includes document ID indexes and node ID indexes over non-partitioned XML table spaces and XML indexes.
2. Includes document ID indexes and node ID indexes over partitioned XML table spaces.

CHECK INDEX does not set a utility restrictive state if the target object is DSNDB01.SYSUTILX.

CHECK INDEX of an XML index cannot run if REBUILD INDEX, REORG INDEX, or RECOVER is being run on that index because CHECK INDEX needs access to the node ID index. CHECK INDEX SHRLEVEL CHANGE cannot run two jobs concurrently for two different indexes that are in the same table space or partition because the snapshot shadow will have a conflicting name for the table space.

Compatibility

The following table shows which utilities can run concurrently with CHECK INDEX on the same target object. The first column lists the other utility and the second column lists whether or not that utility is compatible with CHECK INDEX. The target object can be a table space, an index space, or an index partition. If compatibility depends on particular options of a utility, that information is also documented in the table.

Table 15. Compatibility of CHECK INDEX SHRLEVEL REFERENCE with other utilities

<table>
<thead>
<tr>
<th>Action</th>
<th>Compatible with CHECK INDEX?</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHECK DATA</td>
<td>No</td>
</tr>
<tr>
<td>CHECK INDEX</td>
<td>Yes</td>
</tr>
<tr>
<td>CHECK LOB</td>
<td>Yes</td>
</tr>
<tr>
<td>COPY INDEXSPACE</td>
<td>Yes</td>
</tr>
<tr>
<td>COPY TABLESPACE</td>
<td>Yes</td>
</tr>
<tr>
<td>DIAGNOSE</td>
<td>Yes</td>
</tr>
<tr>
<td>LOAD</td>
<td>No</td>
</tr>
<tr>
<td>MERGECOPY</td>
<td>Yes</td>
</tr>
<tr>
<td>MODIFY</td>
<td>Yes</td>
</tr>
<tr>
<td>QUIESCE</td>
<td>Yes</td>
</tr>
<tr>
<td>REBUILD INDEX</td>
<td>No</td>
</tr>
<tr>
<td>RECOVER INDEX</td>
<td>No</td>
</tr>
<tr>
<td>RECOVER TABLESPACE</td>
<td>No</td>
</tr>
</tbody>
</table>
Table 15. Compatibility of CHECK INDEX SHRLEVEL REFERENCE with other utilities (continued)

<table>
<thead>
<tr>
<th>Action</th>
<th>Compatible with CHECK INDEX?</th>
</tr>
</thead>
<tbody>
<tr>
<td>REORG INDEX</td>
<td>No</td>
</tr>
<tr>
<td>REORG TABLESPACE UNLOAD CONTINUE or PAUSE</td>
<td>No</td>
</tr>
<tr>
<td>REORG TABLESPACE UNLOAD ONLY or EXTERNAL</td>
<td>Yes</td>
</tr>
<tr>
<td>REPAIR DELETE or REPLACE</td>
<td>No</td>
</tr>
<tr>
<td>REPAIR DUMP or VERIFY</td>
<td>Yes</td>
</tr>
<tr>
<td>REPORT</td>
<td>Yes</td>
</tr>
<tr>
<td>RUNSTATS</td>
<td>Yes</td>
</tr>
<tr>
<td>STOSPACE</td>
<td>Yes</td>
</tr>
<tr>
<td>UNLOAD</td>
<td>Yes</td>
</tr>
</tbody>
</table>

To run on SYSIBM.DSNLUX01 or SYSIBM.DSNLUX02, CHECK INDEX must be the only utility within the job step and the only utility that is running in the Db2 subsystem.

**Single logical partitions**

You can run CHECK INDEX on a single logical partition of a secondary index. However, what CHECK INDEX can detect is limited.

- CHECK INDEX does not detect duplicate unique keys in different logical partitions. For example, logical partition 1 might have the following keys:

  A B E F T Z

  Logical partition 2 might have the following keys:

  M N Q T V X

  In this example, the keys are unique within each logical partition, but both logical partitions contain the key, T; so for the index as a whole, the keys are not unique. CHECK INDEX does not detect the duplicates.

- CHECK INDEX does not detect keys that are out of sequence between different logical partitions. For example, the following keys are out of sequence:

  1 7 5 8 9 10 12

  If keys 1, 5, 9, and 12 belong to logical partition 1 and keys 7, 8, and 10 belong to logical partition 2, the keys within each partition are in sequence, but the keys for the index, as a whole, are out of sequence, as shown in the following example:

  LP 1 1 5 9 12 LP 2 7 8 10

  When checking a single logical partition, CHECK INDEX does not detect this out-of-sequence condition.
Indexes in parallel

If you specify more than one index, CHECK INDEX checks the indexes in parallel unless they are constrained by available memory, sort work files, or the PARALLEL option. Sorting the index keys and checking multiple indexes in parallel, rather than sequentially, reduces the elapsed time for a CHECK INDEX job.

If you do not specify the PARALLEL option, the PARAMDEG_UTIL subsystem parameter determines the maximum degree of parallelism for the utility.

The following figure shows the flow of a CHECK INDEX job with a parallel index check for a nonpartitioned table space or a single partition of a partitioned table space.

![Diagram of parallel index check for a nonpartitioned table space or a single partition of a partitioned table space]

Figure 8. Parallel index check for a nonpartitioned table space or a single partition of a partitioned table space

The following figure shows the flow of a CHECK INDEX job with a parallel index check for all partitioning indexes on a partitioned table space.
Figure 9. Parallel index check for all partitioning indexes on a partitioned table space

The following figure shows the flow of a CHECK INDEX job with a parallel index check for a partitioned table space with a single nonpartitioned secondary index.
The following figure shows the flow of a CHECK INDEX job with a parallel index check for all indexes on a partitioned table space. Each unload task pipes keys to each sort task, sorting the keys and piping them back to the check tasks.

Figure 10. Parallel index check for a partitioned table space with a single nonpartitioned secondary index
Reviewing CHECK INDEX output

CHECK INDEX indicates whether a table space and its indexes are inconsistent, but it does not correct any such inconsistencies. If CHECK INDEX detects inconsistencies, you should analyze the output to determine the problem and then correct the inconsistency.

Procedure

To identify the inconsistency:
1. Examine the error messages that CHECK INDEX issues.
2. Verify the point in time for each object that is recovered. Use output from REPORT RECOVERY to ensure that the table space and indexes are recovered to the same point in time. If you specify TOCOPY, TOLASTCOPY, or TOLASTFULLCOPY, ensure that the point in time is a SHRLEVEL REFERENCE copy.
3. If the table space is correct, run the REBUILD INDEX utility to rebuild the indexes.
4. If the index is correct, determine a consistent point in time for the table space, and run the RECOVER utility on the table space. Run CHECK INDEX again to verify consistency.
5. If neither the table space nor its indexes are correct, determine a point in time to which to recover both the table space and indexes, and then rerun the RECOVER utility job, including the table space and its indexes all in the same list.

Related concepts:
Termination or restart of CHECK INDEX

You can terminate and restart the CHECK INDEX utility.

You can terminate CHECK INDEX in any phase without any integrity exposure.

You can restart a CHECK INDEX utility job, but it starts from the beginning again.

Related tasks:
- “Terminating an online utility” on page 43
- “Restarting an online utility” on page 44

Correcting XML data after running CHECK INDEX

After you run the CHECK INDEX utility, you might need to correct XML data.

Procedure

To correct XML data:

Based on the CHECK INDEX output, perform one of the following actions:

<table>
<thead>
<tr>
<th>Problem</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem with a document ID index</td>
<td>1. Confirm that the base table space is at the correct level.</td>
</tr>
<tr>
<td></td>
<td>2. Rebuild the index.</td>
</tr>
<tr>
<td>Problem with an XML table space for a node ID index or an XML index and the index is correct</td>
<td>Run REPAIR LOCATE RID DELETE to remove the orphan row.</td>
</tr>
<tr>
<td>Problem with an XML table space for a node ID index or an XML index and the index is incorrect</td>
<td>Run REBUILD INDEX or RECOVER INDEX to rebuild the index.</td>
</tr>
<tr>
<td>Problem with an XML index over an XML table space</td>
<td>Run REBUILD INDEX to rebuild the index.</td>
</tr>
<tr>
<td></td>
<td><strong>Restriction:</strong> Do not run REPAIR LOCATE RID DELETE to remove orphan rows unless the node ID index does not represent the same row and the base table space does not use the document ID index.</td>
</tr>
</tbody>
</table>

Sample CHECK INDEX control statements

Sample control statements are helpful as models for developing your own CHECK INDEX control statements.
Example 1: Checking all indexes

The control statement specifies that the CHECK INDEX utility is to check all indexes in sample table space DSN8D81A.DSN8S81E.

```
//STEP1 EXEC DSNUPROC,UID='IUIQU1UQ.CHK1',
//       UTPROC='',
//       SYSTEM='DSN'
//SYSERR DD DSN=IUIQU1UQ.CHK3.SYSERR,DISP=(MOD,DELETE,CATLG),
//     UNIT='SYSDA',SPACE=(6000,(20,20),,,ROUND)
//SORTOUT DD DSN=IUIQU1UQ.CHK3.STEP1.SORTOUT,DISP=(MOD,DELETE,CATLG),
//     UNIT='SYSDA',SPACE=(6000,(20,20),,,ROUND)
//SYSIN DD *
CHECK INDEX (ALL) TABLESPACE DSN8D81A.DSN8S81E
/*
```

Example 2: Checking one index

The following control statement specifies that the CHECK INDEX utility is to check the project-number index (DSN8B10.XPROJ1) on the sample project table.

```
SORTDEVT SYSDA specifies that SYSDA is the device type for temporary data sets that are to be dynamically allocated by the sort program.
CHECK INDEX (DSN8B10.XPROJ1)
      SORTDEVT SYSDA
```

Example 3: Checking more than one index

The following control statement specifies that the CHECK INDEX utility is to check the indexes DSN8B10.XEMPRAC1 and DSN8B10.XEMPRAC2 on the employee-to-project-activity sample table.

```
CHECK INDEX NAME (DSN8B10.XEMPRAC1, DSN8B10.XEMPRAC2)
```

Example 4: Checking partitions of all indexes

In the following control statement, table space DB0S0301.TP0S0301 has one partitioned index (ADMF001.IP0S0301), one data-partitioned secondary index (ADMF001.ID0S0302), and one nonpartitioned secondary index (ADMF001.IX0S0303). The (ALL) option indicates that all three indexes on the table space are to be checked. PART 3 indicates that CHECK INDEX is to check the third physical partition of any partitioned indexes and the third logical partition of any nonpartitioned indexes.

```
CHECK INDEX (ALL) TABLESPACE DB0S0301.TPOS0301 PART 3 SORTDEVT SYSDA
```

In this case, CHECK INDEX checks the third partition of ADMF001.IP0S0301, the third partition of ADMF001.ID0S0302, and the third logical partition of ADMF001.IX0S0303, as indicated by the following output.

```
DSNU050I DSNUGUTC- CHECK INDEX(ALL) TABLESPACE DB0S0301.TPOS0301 PART 3 SORTDEVT SYSDA
DSNU700I= DSNUKGET- 10 INDEX ENTRIES UNLOADED FROM INDEX='ADMF001.IP0S0301' PARTITION=3
DSNU700I= DSNUKGET- 10 INDEX ENTRIES UNLOADED FROM INDEX='ADMF001.ID0S0302' PARTITION=3
DSNU701I= DSNUKGET- 10 INDEX ENTRIES UNLOADED FROM 'ADMF001.IX0S0303'
DSNU705I DSNUK001- UNLOAD PHASE COMPLETE - ELAPSED TIME=00:00:00
DSNU717I= DSNUKTER- 10 ENTRIES CHECKED FOR INDEX 'ADMF001.IP0S0301' PARTITION=3
DSNU717I= DSNUKTER- 10 ENTRIES CHECKED FOR INDEX 'ADMF001.ID0S0302' PARTITION=3
DSNU717I= DSNUKTER- 10 ENTRIES CHECKED FOR INDEX 'ADMF001.IX0S0303' PARTITION=3
DSNU720I DSNUK001- CHECKIX PHASE COMPLETE, ELAPSED TIME=00:00:00
DSNU010I DSNUGBAC- UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```

Figure 12. CHECK INDEX output from a job that checks the third partition of all indexes.
Example 5: Checking indexes in a list

The LISTDEF control statement defines a list of indexes called CHKIDXB_LIST. The CHECK INDEX control statement specifies that CHECK INDEX is to check all indexes that are included in the CHKIDXB_LIST list. SORTDEVT SYSDA specifies that SYSDA is the device type for temporary data sets that are to be dynamically allocated by the sort program. SORTNUM 4 specifies that four of these data sets are to be dynamically allocated.

```
//CHKIDXB EXEC PGM=DSNUTILB,REGION=4096K,PARM='SSTR,CHKINDEX1'
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//UTPRINT DD SYSOUT=A
//DSNTRACE DD SYSOUT=A
//SYSOUT DD UNIT=SYSDA,SPACE=(CYL,(5,2)),VOL=SER=SCR03
//SORTLIB DD DISP=SHR,DSN=SYS1.SORTLIB
//SORTOUT DD UNIT=SYSDA,SPACE=(CYL,(5,2)),VOL=SER=SCR03
//SYSERR DD UNIT=SYSDA,SPACE=(CYL,(5,2)),VOL=SER=SCR03
//SYSIN DD *
LISTDEF CHKIDXB LIST INCLUDE INDEXSPACE DBOT55.* ALL
CHECK INDEX LIST CHKIDXB LIST
   SORTDEVT SYSDA
   SORTNUM 4
/*
```

Figure 13. Example of checking indexes in a list

Example 6: Checking all specified indexes on clone tables

The following control statement specifies that the CHECK INDEX utility is to check all specified indexes that are on clone tables.

```
CHECK INDEX (ALL) TABLESPACE DBLOB01.TSLOBC4 CLONE

Related reference:

- [Db2 Sort for z/OS](#)

Related information:

- [DFSORT Application Programming Guide](#)
Chapter 13. CHECK LOB

You can run the CHECK LOB online utility on a LOB table space to identify any structural defects in the LOB table space and any invalid LOB values.

The CHECK LOB utility is useful in a variety of circumstances:
• Run the utility on a LOB table space that is in CHECK-pending (CHKP) status to identify structural defects. If none are found, the CHECK LOB utility turns the CHKP status off.
• Run the utility on a LOB table space that is in auxiliary-warning (AUXW) status to identify invalid LOBs. If none exist, the CHECK LOB utility turns AUXW status off.
• Run the utility after a conditional restart or a point-in-time recovery on all table spaces where LOB table spaces might not be synchronized.
• Run the utility before you run the CHECK DATA utility on a table space that contains at least one LOB column.

Output

After successful execution, CHECK LOB SHRLEVEL CHANGE does not set or reset the CHECK-pending (CHKP) and auxiliary-warning (AUXW) statuses.

If the utility finds any inconsistencies, the LOB table space that is checked is not put into the CHECK-pending status. You can force the prior behavior, that a LOB table space is put into CHECK-pending status when inconsistencies are detected, by specifying CHECK_SETCHKP=Y on the CHECK_SETCHKP system parameter.

Authorization required

To execute this utility, you must use a privilege set that includes one of the following authorities:
• STATS privilege for the database
• DBADM, DBCTRL, or DBMAINT authority for the database. If the object on which the utility operates is in an implicitly created database, DBADM authority on the implicitly created database or DSND804 is required.
• System DBADM authority
• SYSCTRL or SYSADM authority

An ID with installation SYSOPR authority can also execute CHECK LOB.

If you are using SHRLEVEL CHANGE, the user ID that invokes COPY with the CONCURRENT option must provide the necessary authority to execute the DFSMSdss AD md command. DFSMSdss will create a shadow data set with the authority of the utility batch address space. The submitter should have a RACF ALTER authority, or its equivalent, for the shadow data set.

Execution phases of CHECK LOB

The CHECK LOB utility operates in the following phases:
**UTILINIT**
Performs initialization

**CHECKLOB**
Scans all active pages of the LOB table space; generates up to four records per LOB page; passes records to the SORTIN phase

**SORTIN**
Passes CHECKLOB phase records to SORT

**SORT**
Sorts the records from the CHECKLOB phase

**SORTOUT**
Passes sorted records to the REPTLOB phase

**REPTLOB**
Examines records that are produced by the CHECKLOB phase; issues error messages

**UTILTERM**
Performs cleanup

---

**Syntax and options of the CHECK LOB control statement**

The CHECK LOB utility control statement, with its multiple options, defines the function that the utility job performs.

You can create a control statement with the ISPF/PDF edit function. After creating it, save it in a sequential or partitioned data set. When you create the JCL for running the job, use the SYSIN DD statement to specify the name of the data set that contains the utility control statement.

**Syntax diagram**

```
CHECKLOB  lob-table-space-spec  SHRLEVEL—REFERENCE  drain-spec
         SHRLEVEL—CHANGE
       EXCEPTIONS  integer
       EXCEPTIONS 0
       PUNCHDDN—SYSPPUNCH
       PUNCHDDN—ddname
       SORTDEVT—device-type
       SORTNUM—integer

lob-table-space-spec:

TABLESPACE—database-name.  lob-table-space-name  CLONE
```
Option descriptions

LOB
Indicates that you are checking a LOB table space for defects.

TABLESPACE database-name.lob-table-space-name
Specifies the table space to which the data belongs.

database-name is the name of the database and is optional.
The default value is DSNDB04.

lob-table-space-name is the name of the LOB table space.

SHRLEVEL
Indicates the type of access that is to be allowed for the index, table space, or partition that is to be checked during CHECK LOB processing.

REFERENCE
Specifies that applications can read from but cannot write to the index, table space, or partition that is to be checked.

CHANGE
Specifies that applications can read from and write to the index, table space, or partition that is to be checked.

If you specify SHRLEVEL CHANGE, Db2 performs the following actions:

- Drains all writers and forces the buffers to disk for the specified object and all of its indexes
- Invokes DFSMSdss to copy the specified object and all of its indexes to shadow data sets
- Enables read-write access for the specified object and all of its indexes
- Runs CHECK INDEX on the shadow data sets

By default, DFSMSdss uses FlashCopy to copy Db2 objects to shadow data sets, if FlashCopy is available. If DFSMSdss cannot use FlashCopy, DFSMSdss uses a slower method. As a result, creating copies of objects might take a long time, and the time during which the data and indexes have read-only access might increase. You can set the CHECK_FASTREPLICATION subsystem parameter to REQUIRED to force the CHECK utility to use only FlashCopy. If FlashCopy is not available, the CHECK utility fails.

Related information:
“Subsystem parameters for refining DFSMSdss COPY operation with utilities” on page 40

DRAIN_WAIT
Specifies the number of seconds that CHECK LOB is to wait when draining the table space or index. The specified time is the aggregate time for objects that
are to be checked. This value overrides the values that are specified by the
IRLMRWT and UTIMOUT subsystem parameters.

integer can be any integer from 0 to 1800. If you do not specify DRAIN_WAIT
or specify a value of 0, CHECK LOB uses the value of the lock timeout
subsystem parameter IRLMRWT.

RETRY integer
Specifies the maximum number of retries that CHECK LOB is to attempt.

integer can be any integer from 0 to 255. If you do not specify RETRY, CHECK
LOB uses the value of the utility multiplier system parameter UTIMOUT.

Specifying RETRY can increase processing costs and result in multiple or
extended periods during which the specified index, table space, or partition is
in read-only access.

RETRY_DELAY integer
Specifies the minimum duration, in seconds, between retries. integer can be any
integer from 1 to 1800.

If you do not specify RETRY_DELAY, CHECK LOB uses the smaller of the
following two values:

- DRAIN_WAIT value × RETRY value
- DRAIN_WAIT value × 10

EXCEPTIONS integer
Specifies the maximum number of exceptions, which are reported by messages
only. CHECK LOB terminates in the CHECKLOB phase when it reaches the
specified number of exceptions.

All defects that are reported by messages are applied to the exception count.

integer is the maximum number of exceptions.

The default value is 0, which indicates no limit on the number of exceptions.

PUNCHDDN ddname
Specifies the DD statement for a data set that is to receive the REPAIR utility
control statements that CHECK LOB SHRLEVEL CHANGE generates. The
REPAIR statements generated deletes the LOBs reported in error messages
from the LOB table space. CHECK DATA should then be run against the base
table space to set the deleted LOB columns in the base records to invalid.

ddname is the DD name.

The default value is SYSPUNCH.

The PUNCHDDN keyword specifies either a DD name or a TEMPLATE name
specification from a previous TEMPLATE control statement. If utility
processing detects that the specified name is both a name in the current job
step and a TEMPLATE name, the utility uses the DD name.

SORTDEVT device-type
Specifies the device type for temporary data sets that are to be dynamically
allocated by the sort program.

A TEMPLATE specification does not dynamically allocate sort work data sets. The
SORTDEVT keyword controls dynamic allocation of these data sets.

device-type is the device type and can be any disk device type that is acceptable
to the DYNALLOC parameter of the SORT or OPTION control statement for
the sort program. Tape devices are not supported by the sort program.
If you omit SORTDEVT and a sort is required, you must provide the DD statements that the sort program requires for the temporary data sets.

**SORTNUM integer**

Indicates the number of temporary data sets that are to be dynamically allocated by the sort program.

*integer* is the number of temporary data sets that can range from 2 to 255.

If you omit SORTDEVT, SORTNUM is ignored. If you use SORTDEVT and omit SORTNUM, no value is passed to the sort program, which then uses its own default. You need at least two sort work data sets for each sort.

**Important:** The SORTNUM keyword is ignored if the UTSORTAL subsystem parameter is set to YES and the IGNSORTN subsystem parameter is set to YES.

**Clone**

Indicates that CHECK LOB is to check the LOB space data for only the clone table, not the LOB data for the base table.

**Related reference:**

[Db2 Sort for z/OS](#)

**Related information:**

[DFSORT Application Programming Guide](#)

---

### Before running CHECK LOB

Certain activities might be required before you run the CHECK LOB utility, depending on your situation.

You must first recover a LOB table space that is in RECOVER-pending status before running CHECK LOB.

Beginning in Version 8, the CHECK LOB utility does not require SYSUT1 and SORTOUT data sets. Work records are written to and processed from an asynchronous SORT phase. The WORKDDN keyword, which provided the DD names of the SYSUT1 and SORTOUT data sets in earlier versions of Db2, is not needed and is ignored. You do not need to modify existing control statements to remove the WORKDDN keyword.

### Data sets that CHECK LOB uses

The CHECK LOB utility uses a number of data sets during its operation.

The following table lists the data sets that CHECK LOB uses. The table lists the DD name that is used to identify the data set, a description of the data set, and an indication of whether it is required. Include statements in your JCL for each required data set and any optional data sets that you want to use.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSIN</td>
<td>An input data that contains the utility control statement.</td>
<td>Yes</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>An output data set for messages.</td>
<td>Yes</td>
</tr>
<tr>
<td>UTPRINT</td>
<td>A data set that contains messages from the sort program (usually, SYSOUT or DUMMY).</td>
<td>Yes</td>
</tr>
</tbody>
</table>
The following object is named in the utility control statement and does not require DD statements in the JCL:

**Table space**
Object that is to be checked.

Sort work data sets cannot span volumes. Smaller volumes require more sort work data sets to sort the same amount of data; therefore, large volume sizes can reduce the number of needed sort work data sets. When you allocate sort work data sets on disk, the recommended amount of space to allow provides at least 1.2 times the amount of data that is to be sorted.

Tape devices are not supported for sort work data sets.

**Related concepts:**
Data sets that online utilities use” on page 16

**Related reference:**
- Db2 Sort for z/OS

**Related information:**
- DFSORT Application Programming Guide

**Shadow data sets for CHECK LOB**
When you execute the CHECK LOB utility with the SHRLEVEL CHANGE option, the utility uses shadow data sets.

If a table space, partition, or index resides in Db2-managed data sets and shadow data sets do not already exist when you execute CHECK LOB, Db2 creates the shadow data sets. At the end of CHECK LOB processing, the Db2-managed shadow data sets are deleted.

For user-managed data sets, DFSMSdss can create or scratch and recreate the required shadow data sets as needed. When the CHECK LOB utility completes the processing of user-managed data sets, the shadow data sets are not automatically scratched.

If you have not changed the value of FREEPAGE or PCTFREE on the CREATE TABLESPACE statement, the amount of required space for a shadow data set is comparable to the amount of required space for the original data set.

If you do not want the shadow data sets to be allocated in the same storage class as the production data sets, set the UTIL_TEMP_STORCLAS system parameter to specify the storage class for the shadow data sets.

**Shadow data set names**
Each shadow data set must have the following name:

```
catname.DSNDBx.dbname.psname.y000z.Lnnn
```

In the preceding name, the variables have the following meanings:

**variable**
**meaning**
**catname**
The VSAM catalog name or alias

**x**
C or D

**dbname**
Database name

**psname**
Table space name or index name

**y**
I or J

**z**
1 or 2

**Lnnn**
Partition identifier. Use one of the following values:
- A001 through A999 for partitions 1 through 999
- B000 through B999 for partitions 1000 through 1999
- C000 through C999 for partitions 2000 through 2999
- D000 through D999 for partitions 3000 through 3999
- E000 through E996 for partitions 4000 through 4096

To determine the names of existing data sets, execute one of the following queries against the SYSTABLEPART or SYSINDEXPART catalog tables:

```sql
SELECT DBNAME, TSNAME, IPREFIX
FROM SYSCB.SYSTABLEPART
WHERE DBNAME = 'dbname'
AND TSNAME = 'psname';

SELECT DBNAME, IXNAME, IPREFIX
FROM SYSCB.SYSINDEXES X, SYSCB.SYSINDEXPART Y
WHERE X.NAME = Y.IXNAME
AND X.CREATOR = Y.IXCREATOR
AND X.DBNAME = 'dbname'
AND X.INDEXSPACE = 'psname';
```

**Defining shadow data sets**

For a partitioned table space, Db2 returns rows from which you select the row for the partitions that you want to check.

Consider the following actions when you preallocate the data sets:
- Allocate the shadow data sets according to the rules for user-managed data sets.
- Define the shadow data sets as LINEAR.
- Use SHAREOPTIONS(3,3).
- Define the shadow data sets as EA-enabled if the original table space or index space is EA-enabled.
- Allocate the shadow data sets on the volumes that are defined in the storage group for the original table space or index space.

If you specify a secondary space quantity, Db2 does not use it. Instead, Db2 uses the SECQTY value for the table space or index space.
**Recommendation:** Use the MODEL option, which causes the new shadow data set to be created like the original data set. This method is shown in the following example:

```sql
DEFINE CLUSTER +
  (NAME('catname.DSNDBC.dbname.psname.x000z.L001') +
   MODEL('catname.DSNDBC.dbname.psname.y000z.L001')) +
DATA +
  (NAME('catname.DSNDBD.dbname.psname.x000z.L001') +
   MODEL('catname.DSNDBD.dbname.psname.y000z.L001'))
```

**Creating shadow data sets for indexes**

When you preallocate shadow data sets for indexes, create the data sets as follows:

- Create shadow data sets for the partition of the table space and the corresponding partition in each partitioning index and data-partitioned secondary index.
- Create a shadow data set for logical partitions of nonpartitioned secondary indexes.

Use the same naming scheme for these index data sets as you use for other data sets that are associated with the base index, except use J0001 instead of I0001.

**Concurrency and compatibility for CHECK LOB**

The CHECK LOB utility has certain concurrency and compatibility characteristics associated with it.

Db2 treats individual data and index partitions as distinct target objects. Utilities that operate on different partitions of the same table space or index space are compatible.

**Claims and drains**

The following table shows which claim classes CHECK LOB claims and drains and any restrictive state that the utility sets on the target object.

<table>
<thead>
<tr>
<th>Target objects</th>
<th>CHECK LOB SHRLLEVEL REFERENCE</th>
<th>CHECK LOB SHRLLEVEL CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOB table space</td>
<td>DW/UTRO</td>
<td>CR/UTRW</td>
</tr>
<tr>
<td>Index on the auxiliary table</td>
<td>DW/UTRO</td>
<td>CR/UTRW</td>
</tr>
</tbody>
</table>

**Legend:**
- CR: Claim the read claim class
- DW: Drain the write claim class, concurrent access for SQL readers
- UTRO: Utility restrictive state, read-only access allowed
- UTRW: Utility restrictive state, read and write access allowed

**Compatibility**

Any SQL operation or other online utility that attempts to update the same LOB table space is incompatible.
How CHECK LOB identifies violations

You can find and resolve violations by reviewing messages that the CHECK LOB utility issues.

CHECK LOB issues message DSNU743I whenever it finds a LOB value that is invalid. The violation is identified by the row ID and version number of the LOB. You can resolve LOB violations by using the UPDATE or DELETE SQL statements to update the LOB column or delete the row that is associated with the LOB. (Use the row ID from message DSNU743I.)

Contact IBM Support for assistance with diagnosing and resolving the problem.

Related reference:
- DELETE (Db2 SQL)
- UPDATE (Db2 SQL)

Removing CHECK-pending status for a LOB table space

If a LOB table space has a status of CHECK-pending, you can remove the CHECK-pending status by correcting the error and either rerunning the CHECK LOB utility with SHRLEVEL REFERENCE specified or by running the REPAIR utility.

About this task

Note: The CHECK LOB utility sets or resets the CHECK-pending status when errors are found only the CHECK_SETCHKP subsystem parameter is set to YES.

Procedure

To remove CHECK-pending status:
1. Correct any defects that are found in the LOB table space by using the REPAIR utility.

   Attention: Use the REPAIR utility with care because improper use can further damage the data. If necessary, contact IBM Support for guidance on using the REPAIR utility.

2. Run CHECK LOB again, or run the REPAIR utility to reset CHECK-pending or auxiliary-warning status.

Related reference:
- “Syntax and options of the CHECK LOB control statement” on page 122
- Chapter 29, “REPAIR,” on page 669
- SET CHECK PENDING field (CHECK_SETCHKP subsystem parameter) (Db2 Installation and Migration)
- “CHECK-pending status” on page 1039

Resolving media failure

Some media failures leave LOB pages in the logical page list (LPL), which requires action.
Procedure

To resolve media failure:

Run CHECK LOB on a LOB table space. The pages that were in the LPL are removed from the list so that they are available.

Related tasks:

[Displaying the logical page list (Db2 Administration Guide)]

Termination or restart of CHECK LOB

You can terminate and restart the CHECK LOB utility.

If you terminate CHECK LOB during the CHECKLOB phase, LOB table spaces remain in CHECK-pending status. During normal execution, the CHECKLOB phase places the LOB table space in CHECK-pending status; at the end of the phase, the CHECK-pending status is reset if no errors are detected.

You can restart a CHECK LOB utility job, but it starts from the beginning again.

Related tasks:

“Terminating an online utility” on page 43
“Restarting an online utility” on page 44

Sample CHECK LOB control statements

Sample control statements are helpful as models for developing your own CHECK LOB control statements.

Example 1: Checking a LOB table space

The following control statement specifies that the CHECK LOB utility is to check LOB table space DBIQUG01.TLIQUG02 for structural defects or invalid LOB values. The EXCEPTIONS 3 option indicates that the CHECK LOB utility is to terminate when it finds three exceptions. The SORTDEVT and SORTNUM options provide information about temporary data sets that are to be dynamically allocated by the sort program. SORTDEVT SYSDA specifies that the device type is SYSDA, and SORTNUM 4 indicates that four temporary data sets are to be dynamically allocated by the sort program.

```
//STEP1 EXEC DSNUPROC,UID='IUIQU2UG.CHECKL',
  // UTPROC='',
  // SYSTEM='DSN'
//SYSIN DD *
CHECK LOB TABLESPACE DBIQUG01.TLIQUG02
  EXCEPTIONS 3 SORTDEVT SYSDA
  SORTNUM 4
```

Example 2: Checking the LOB table space data

The following control statement specifies that the CHECK LOB utility is to check the LOB table space data with the SHRLEVEL CHANGE option, which specifies that the application can read from and write to the table space that is to be checked.

```
//STEP2 EXEC DSNUPROC,
  // UTPROC='',SYSTEM='SSTR',
  // UID='CHKLOB12.STEP2'
```
Example 3: Checking the LOB space data for a clone table

The following control statement specifies that the CHECK LOB utility is to check the LOB space data for only the clone table, not the LOB data for the base table. The EXCEPTIONS 0 option indicates that there is no limit on the number of exceptions. The SORTDEVT and SORTNUM options provide information about temporary data sets that are to be dynamically allocated by the sort program. SORTDEVT SYSDA specifies that the device type is SYSDA, and SORTNUM 10 indicates that ten temporary data sets are to be dynamically allocated by the sort program.

CHECK LOB TABLESPACE DBA12.TSL12  CLONE
EXCEPTIONS 5

Related reference:

Db2 Sort for z/OS

Related information:

DFSORT Application Programming Guide
Chapter 14. COPY

The COPY online utility creates copies of certain objects. These copies, called image copies, can later be used for recovery.

COPY can create up to five image copies: two sequential image copies for the local site, two sequential image copies for the recovery site, and one FlashCopy image copy. These copies can be created for any of the following objects:
- Table space
- Table space partition
- Data set of a nonpartitioned table space
- Index space
- Index space partition

The sequential image copies can be either full or incremental. A full image copy is a copy of all pages in a table space, partition, data set, or index space. An incremental image copy is a copy of the system pages and only those data pages that have been modified since the last use of the COPY utility.

The RECOVER utility uses image copies when recovering a table space or index space to the most recent time or to a previous point in time. Copies can also be used by the MERGECOPY, COPYTOCOPY, and UNLOAD utilities.

Output

The COPY utility produces up to four sequential data sets that contain the image copy and optionally one FlashCopy image copy. COPY also adds rows in the SYSIBM.SYSCOPY catalog table that describe the image copy data sets that are available to the RECOVER utility. Your installation is responsible for ensuring that these data sets are available if the RECOVER utility requests them.

COPY resets COPY-pending status as follows:
- If the copy is a full image copy, Db2 resets any COPY-pending status for the copied table spaces.
- If you copy a single table space partition, Db2 resets any COPY-pending status only for the copied partition and not for the whole table space.
- If you copy a single piece of a multi-piece linear data set, Db2 does not reset any COPY-pending status.
- If you copy an index space or index, Db2 resets any informational COPY-pending (ICOPY) status.
- If you copy a NOT LOGGED table space, Db2 resets any informational COPY-pending (ICOPY) status.

Authorization required

To execute this utility, you must use a privilege set that includes one of the following authorities:
- IMAGCOPY privilege for the database
• DBADM, DBCTRL, or DBMAINT authority for the database. If the object on which the utility operates is in an implicitly created database, DBADM authority on the implicitly created database or DSNDB04 is required.
• System DBADM authority
• SYSCTRL or SYSADM authority

An ID with installation SYSOPR authority can also execute COPY, but only on a table space in the DSNDB01 or DSNDB06 database.

If the CONCURRENT option is specified, the user ID that invokes the COPY utility must have the authority to execute the DFSMSdss DUMP command.

If either the FLASHCOPY YES or FLASHCOPY CONSISTENT option is specified, the user ID that invokes the COPY utility must have the authority to execute the DFSMSdss COPY command.

**Restrictions on running COPY**

COPY cannot be run on a table space during the period after RECOVER is run to a point in time before materialization of pending definition changes and before REORG is run to complete the point-in-time recovery process.

**Execution phases of COPY**

The COPY utility operates in the following phases:

**UTILINIT**
Performs initialization and setup.

**REPORT**
Reports image copy information if the CHANGELIMIT option is specified.

**COPY**
Creates copies.

If FLASHCOPY YES or FLASHCOPY CONSISTENT is specified, the FlashCopy image copies are created in this phase. Otherwise, sequential image copies are created in this phase.

**SEQCOPY**
Creates additional sequential image copies from a FlashCopy image copy if FLASHCOPY YES or FLASHCOPY CONSISTENT is specified. The phase is included only when both FlashCopy image copies and sequential image copies are requested.

**LOGAPPLY**
Identifies the most recent checkpoint for each member. All objects that are being copied are updated to the same log point to prepare for backout processing.

If FLASHCOPY CONSISTENT is specified, Db2 applies the updates to the FlashCopy image copy to ensure that copy includes all activity up to the point of consistency.

**LOGCSR**
Calls log apply do the current status rebuild (CSR).

If FLASHCOPY CONSISTENT is specified, the utility reads the logs during this phase. The utility uses the logs to identify the uncommitted work that needs to be backed out of the image copy.
LOGUNDO
Backs out uncommitted work from the image copy.
If FLASHCOPY CONSISTENT is specified, the utility backs out uncommitted work from the FlashCopy image copy to make the image copy consistent.

UTILTERM
Performs cleanup.

Related reference:
Chapter 26, “RECOVER,” on page 459

Syntax and options of the COPY control statement
The COPY utility control statement, with its multiple options, defines the function that the utility job performs.

You can create a control statement with the ISPF/PDF edit function. After creating it, save it in a sequential or partitioned data set. When you create the JCL for running the job, use the SYSIN DD statement to specify the name of the data set that contains the utility control statement.

Syntax diagram

Notes:
1 Use the copy-spec if you do not want to use the CONCURRENT option.
2 Use the concurrent-spec if you want to use the CONCURRENT option, but not the FILTERDDN option.
3 Use the filterddn spec if you want to use the CONCURRENT and FILTERDDN options.

copy-spec:
Notes:
1  Not valid for nonpartitioning indexes.
2  CHECKPAGE is the default for table spaces.

concurrent-spec:

Notes:
1  Not valid for nonpartitioning indexes.

filterddn-spec:

Notes:
1  Not valid for nonpartitioning indexes.
data-set-spec:

```
COPYDDN(ddname1, ddname2), ddname2
RECOVERYDDN(ddname3, ddname4)
```

Notes:
1. COPYDDN SYSCOPY is the default for the primary copy, but this default can only be used for one object in the list.

changelog-spec:

```
CHANGELIMIT(ANY)
(REPORTONLY(percent_value1, percent_value2))
```

ts-spec:

```
TABLESPACE table-space-name
```  

index-spec:

```
INDEXSPACE database-name, index-space-name
INDEX(index-name, creator-id)
```

Notes:
1. INDEXSPACE is the preferred specification.

Option descriptions

**LIST listdef-name**

Specifies the name of a previously defined LISTDEF list name. LIST specifies
one LIST keyword for each COPY control statement. Do not specify LIST with either the INDEX or the TABLESPACE keyword. Db2 invokes COPY once for the entire list.

This utility does not support lists that specify more than 32,000 objects.

Partitions of table spaces or index spaces that are included by the PARTLEVEL keyword count as separate objects.

This utility processes clone data only if the CLONE keyword is specified. The use of CLONED YES on the LISTDEF statement is not sufficient.

The partitions or partition ranges can be specified in a list.

**TABLESPACE** database-name.table-space-name

Specifies the table space (and, optionally, the database it belongs to) that is to be copied.

database-name is the name of the database that the table space belongs to. The default value is DSNDB04.

Table-space-name is the name of the table space to be copied.

Specify the DSNDB01.SYSUTILX, DSNDB06.SYSTSCPY, or DSNDB01.SYSLGRNX table space by itself in a single COPY statement.

Alternatively, specify the DSNDB01.SYSUTILX, DSNDB06.SYSTSCPY, or DSNDB01.SYSLGRNX table space with indexes over the table space that were defined with the COPY YES attribute.

If you specify a segmented table space, COPY locates empty and unformatted data pages in the table space and does not copy them.

You cannot copy a table space that uses a storage group that is defined with a mixture of specific and non-specific volume IDs.

**CLONE**

Indicates that COPY is to copy only clone table or index data. This utility will only process clone data if the CLONE keyword is specified. The use of CLONED YES on the LISTDEF statement is not sufficient.

If the utility is processing a table space and CLONE is specified, the utility will only process clone table data. If the utility is processing an index and CLONE is specified, the utility will only process indexes over clone tables. If you use the LIST keyword to specify a list of objects, COPY processes only those objects in the list that contain clone tables or indexes on clone tables. COPY ignores other objects in the list.

**INDEXSPACE** database-name.index-space-name

Specifies the qualified name of the index space that is to be copied; the name is obtained from the SYSIBM.SYSINDEXES table. The specified index space must be defined with the COPY YES attribute.

database-name Optionally specifies the name of the database that the index space belongs to. The default value is DSNDB04.

index-space-name specifies the name of the index space that is to be copied.

You cannot copy an index space that uses a storage group that is defined with mixture of specific and non-specific volume IDs.

**INDEX** creator-id.index-name

Specifies the index that is to be copied. Enclose the index name in quotation marks if the name contains a blank.
creator-id optionally specifies the creator of the index. The default value is the user identifier for the utility.

index-name specifies the name of the index that is to be copied.

COPYDDN (ddname1,ddname2)
Specifies a DD name or a TEMPLATE name for the primary (ddname1) and backup (ddname2) copy data sets for the image copy at the local site.

You can use the COPYDDN keyword to specify either a DD name or a TEMPLATE name specification from a previous TEMPLATE control statement. If utility processing detects that the specified name is both a DD name in the current job step and a TEMPLATE name, the utility uses the DD name. For more information about TEMPLATE specifications, see Chapter 34, "TEMPLATE," on page 793.

dname is the DD name. The default value is SYSCOPY for the primary copy. You can use the default for only one object in the list. The first object in the list that does not have COPYDDN specified uses the default. Any other objects in the list that do not have COPYDDN specified cause an error.

If you use the CHANGELIMIT REPORTONLY option, you can use a DD DUMMY statement when you specify the SYSCOPY output data set. This card prevents a data set from being allocated and opened.

Recommendation: Catalog all of your image copy data sets.

You cannot have duplicate image copy data sets. If the DD statement identifies a noncataloged data set with the same name, volume serial, and file sequence number as one that is already recorded in the SYSIBM.SYSCOPY catalog table, the COPY utility issues a message and does not make an image copy. If COPY identifies a cataloged data set with only the same name, it does not make an image copy. For cataloged image copy data sets, CATLG must be specified for the normal termination disposition in the DD statement, as shown in the following example:

```
DISP=(MOD,CATLG,CATLG)
```

The DSVOLSER field of the SYSCOPY entry is blank.

If you use the CONCURRENT and FILTERDDN options, ensure that the size of the copy data set is large enough to include all of the objects in the list.

RECOVERYDDN (ddname3,ddname4)
Specifies a DD name or a template name for the primary (ddname3) and backup (ddname4) copy data sets for the image copy at the recovery site.

You can use the RECOVERYDDN keyword to specify either a DD name or a template name. If utility processing detects that the specified name is both a DD name in the current job step and a template name, the utility THE uses the DD name.

dname3 and dname4 are DD names.

You cannot have duplicate image copy data sets.

If you use the CONCURRENT and FILTERDDN options, ensure that the size of the copy data set is large enough to include all of the objects in the list.

FULL
Specifies that COPY is to make either a full or an incremental image copy.
YES
Specifies a full image copy. Making a full image copy resets the COPY-pending status for the table space or index, or for the partition if you specify DSNUM.

NO
Specifies only an incremental image copy. Only changes since the last image copy are to be copied. NO is not valid for indexes.

Incremental image copies are not allowed in the following situations:
- The last full image copy of the table space was taken with the CONCURRENT option.
- No full image copies exist for the table space or data set that is being copied.
- After a successful LOAD or REORG operation, unless an inline copy was made during the LOAD or REORG job.
- You specify one of the following table spaces: DSNDB01.DBD01, DSNDB01.SYSUTILX, DSNDB06.SYSTSCPY, or DSNDB01.SYSDBDXA.
- A previous COPY was terminated with the -TERM UTIL command, so the most recent SYSIBM.SYSCOPY record for the object contains ICTYPE = T.
- If you specify both FLASHCOPY YES or CONSISTENT and FULL NO, the COPY utility issues an informational message and creates a FlashCopy image copy. (FlashCopy image copies are created as data set level copies of the object and cannot be incremental.) If you also request that sequential image copies be taken, those copies are created from the FlashCopy image copy.

For incremental image copies of partitioned table spaces, COPY includes the header page for each partition that has changed pages.

COPY automatically takes a full image copy of a table space if you specify FULL NO when an incremental image copy is not allowed.

Related information:
“Allocation of sequential image copy data sets” on page 169

CHANGELIMIT
The CHANGELIMIT option is deprecated, and the alternative is running DSNACCOX.

Specifies the limit of changed pages in the table space, partition, or data set at which an incremental or full image copy is to be taken.

ANY
Specifies that COPY is to take a full image copy if any pages have changed since the last image copy.

percent_value1
Specifies the first value in the CHANGELIMIT range. percent_value1 must be an integer or decimal value from 0.0 to 100.0. You do not need to specify leading zeroes, and the decimal point is not required when specifying a whole integer. Specify a maximum of one decimal place for a decimal value. For example, you can specify .5. If you specify this value, COPY takes an incremental image copy if more than one half of one percent of the pages have changed.

percent_value2
Specifies the second value in the CHANGELIMIT range. percent_value2
must be an integer or decimal value from 0.0 to 100.0. You do not need to specify leading zeroes, and the decimal point is not required when specifying a whole integer. Specify a maximum of one decimal place for a decimal value (for example, .5).

COPY CHANGELIMIT accepts percentage values in any order. For example, you can specify (10,1) or (1,10).

If only one percentage value is specified, COPY CHANGELIMIT:

- Creates an incremental image copy if the percentage of changed pages is greater than 0 and less than \textit{percent\_value1}.
- Creates a full image copy if the percentage of changed pages is greater than or equal to \textit{percent\_value1}, or if CHANGELIMIT(0) is specified.
- Does not create an image copy if no pages have changed, unless CHANGELIMIT(0) is specified.
- Always creates a full image copy, even when no pages have been updated since the last image copy, if CHANGELIMIT(0) is specified.
- Creates a full image copy if CHANGELIMIT(100) is specified and all pages have been changed since the last image copy.
- Creates an incremental image copy if CHANGELIMIT(100) is specified and some but not all pages have been changed since the last image copy.

If two percentage values are specified, COPY CHANGELIMIT:

- Creates an incremental image copy if the percentage of changed pages is greater than the lowest specified value and less than the highest specified value.
- Creates a full image copy if the percentage of changed pages is equal to or greater than the highest specified value.
- Does not create an image copy if the percentage of changed pages is less than or equal to the lowest specified value.
- If both values are equal, creates a full image copy if the percentage of changed pages is equal to or greater than the specified value.

The default value is (10).

You cannot use the CHANGELIMIT option for a table space or partition that is defined with TRACKMOD NO. If you change the TRACKMOD option from NO to YES, you must take an image copy before you can use the CHANGELIMIT option. For nonpartitioned table spaces, you must copy the entire table space to allow future CHANGELIMIT requests.

**Related information:**

"Allocation of sequential image copy data sets" on page 169

REPORTONLY

Specifies that image copy information is to be displayed. If you specify the REPORTONLY option, only image copy information is displayed. Image copies are not taken in this case; they are only recommended.

DSNUM

For a table space, identifies a partition or data set within the table space to be copied; or it copies the entire table space. For an index space, DSNUM identifies a partition to be copied, or it copies the entire index space. This option can specify a partition of a data-partitioned secondary index if the index is copy-enabled.
If a data set of a nonpartitioned table space is in the COPY-pending status, you must copy the entire table space.

If DSNUM ALL is implicitly or explicitly specified for a table space that has a partition in PRO restricted status, COPY fails. If COPY is specified for a single partition that is in PRO restricted status, an informational message is issued, and no image copy is produced.

**ALL**

Indicates that the entire table space or index space is to be copied. You must use ALL for a nonpartitioned secondary index.

**integer**

Is the number of a partition or data set that is to be copied.

An integer value is not valid for nonpartitioned secondary indexes.

**For a partitioned table space or index space,** the integer is its physical partition number. The maximum is 4096.

**For a nonpartitioned table space,** find the integer at the end of the data set name as it is cataloged in the ICF catalog. The data set name has the following format:

```
catname.DSNDBx.dbname.spacename.y000Z.Annn
```

In this format:

- **catname** is the ICF catalog name or alias.
- **x** is C (for VSAM clusters) or D (for VSAM data components).
- **dbname** is the database name.
- **spacename** is the table space or index space name.
- **y** is I or J, which indicates the data set name used by REORG with FASTSWITCH.
- **z** is 1 or 2.
- **nnn** is the data set integer.

If COPY takes an image copy of data sets (rather than on table spaces), RECOVER, MERGECOPY, or COPYTOCOPY must use the copies on a data set level. For a nonpartitioned table space, if COPY takes image copies on data sets and you run MODIFY RECOVERY with DSNUM ALL, the table space is placed in COPY-pending status if a full image copy of the entire table space does not exist.

**PARALLEL**

For sequential format image copies, specifies the maximum number of objects in the list that are to be processed in parallel. The utility processes the list of objects in parallel for image copies being written to or from different disk or tape devices. If you specify TAPEUNITS with PARALLEL, you control the number of tape drives that are dynamically allocated for the copy.

If you omit PARALLEL, one object is copied at a time.

**Restriction:** Do not specify the PARALLEL keyword if one or more of the output data sets are defined with DD statements that specify UNIT=AFF to
refer to the same device as a previous DD statement. This usage is not supported with the PARALLEL keyword and could result in an abend. Instead, consider using templates to define your data sets.

The PARALLEL keyword is ignored for FlashCopy image copies.

(num-objects)
   Specifies the number of objects in the list that are to be processed in parallel. You can adjust this value to a smaller value if COPY encounters storage constraints.

   If you specify 0 or do not specify a value for num-objects, COPY determines the optimal number of objects to process in parallel.

TAPEUNITS
   Specifies the maximum number of tape drives that the utility dynamically allocates for the list of objects to be processed in parallel. TAPEUNITS applies only to tape drives that are dynamically allocated through the TEMPLATE keyword. It does not apply to JCL allocated tape drives. The total number of tape drives allocated for the COPY request is the sum of the JCL allocated tape drives plus the number of tape drives determined as follows:

   • the value that is specified for TAPEUNITS
   • The value determined by the COPY utility if you omit the TAPEUNITS keyword

   If you omit this keyword, the utility determines the number of tape drives to dynamically allocate for the copy function.

   The TAPEUNITS keyword is ignored for FlashCopy image copies.

(num-tape-units)
   Specifies the number of tape drives to allocate. If you specify 0 or do not specify the TAPEUNITS keyword, COPY determines the maximum number of tape drives to be dynamically allocated for the function. COPY TAPEUNITS has a max value of 32767.

CHECKPAGE
   Indicates that each page in the table space or index space is to be checked for validity. The validity checking operates on one page at a time and does not include any cross-page checking. If it finds an error, COPY issues a message that describes the type of error. If more than one error exists in a given page, only the first error is identified. COPY continues checking the remaining pages in the table space or index space after it finds an error. CHECKPAGE is the default for table spaces. CHECKPAGE is not the default for indexes. This keyword is ignored by FlashCopy.

   Important: Use of the CHECKPAGE option for indexes can result in greatly increased processor usage.

NOCHECKPAGE
   Indicates that only basic checks on each page in the table space or index space are done. Among the basic checks that are performed are:
   • Whether the page number is valid
   • Whether pages are in the correct order
   • Whether the page is logically broken

SYSTEMPAGES
   Specifies whether the COPY utility puts system pages at the beginning of the image copy data set.
Although the system pages are located at the beginning of many image copies, this placement is not guaranteed. In many cases, the system pages are not included. For example, incremental copies do not include system pages. This keyword is ignored by FlashCopy.

**YES**

Ensures that any header, dictionary, and version system pages are copied at the beginning of the image copy data set. Dictionary and version system pages can be copied twice.

Selecting YES ensures that the image copy contains the necessary system pages for subsequent UNLOAD utility jobs to correctly format and unload all data rows.

**NO**

Does not ensure that the dictionary and version system pages are copied at the beginning of the image copy data set. The COPY utility copies the pages in the current order, including the header pages.

**FLASHCOPY**

Specifies whether FlashCopy technology is used to create a copy of the object. Valid values are YES, NO, or CONSISTENT. When FlashCopy is used, a separate data set is created for each partition or piece of the object.

Specify YES or CONSISTENT only if the Db2 data sets are on FlashCopy Version 2 disk volumes.

The FlashCopy specifications on the utility control statement override any specifications for FlashCopy that are defined by using the Db2 subsystem parameters. If the FlashCopy subsystem parameters specify the use of FlashCopy as the default behavior of this utility, the FLASHCOPY option can be omitted from the utility control statement.

**Important:** If the input data set is less than one cylinder, FlashCopy technology might not be used for copying the objects regardless of the FLASHCOPY settings. The copy is performed by IDCAMS if FlashCopy is not used.

**NO**

Specifies that no FlashCopy is made. NO is the default value for FLASHCOPY.

**YES**

Specifies that FlashCopy technology is used to copy the object.

**Important:** Under the following circumstances, the COPY utility might not use FlashCopy even though YES is specified:

- FlashCopy Version 2 disk volumes are not available
- The source tracks are already the target of a FlashCopy operation
- The target tracks are the source of a FlashCopy operation
- The maximum number of relationships for the copy is exceeded

In the event that FlashCopy is not used, the COPY utility uses traditional I/O methods to copy the object, which can result in longer than expected execution time.

**CONSISTENT**

Specifies that when SHRLEVEL CHANGE is also specified, FlashCopy technology is used to copy the object and that any uncommitted work included in the copy is backed out of the copy to make the copy consistent. If SHRLEVEL CHANGE is not specified, specifying FLASHCOPY CONSISTENT is the same as specifying FLASHCOPY YES.
Specifying FLASHCOPY CONSISTENT requires additional time and system resources during utility processing, because the COPY utility must read the logs and apply the changes to the image copy. Similarly, recovering from a consistent FlashCopy image copy also requires additional time and system resources to read the logs and reapply work that was previously backed out.

**Restriction:** CONSISTENT cannot be specified when copying objects that have been defined with the NOT LOGGED attribute. If CONSISTENT is specified for an object that is defined with the NOT LOGGED attribute, the COPY utility does not make a copy of the object and issues message DSNU076I with return code 8.

**Related information:**
- “FlashCopy image copies” on page 157
- “Subsystem parameters for refining DFSMSdss COPY operation with utilities” on page 40

**FCCOPYDDN**
Specifies the template to be used to create the FlashCopy image copy data set names. If a value is not specified for FCCOPYDDN on the COPY control statement when FlashCopy is used, the value specified on the FCCOPYDDN subsystem parameter determines the template to be used.

```
(template-name)
```

The data set names for the FlashCopy image copy are allocated according to the template specification. For table space or index space level FlashCopy image copies, because a data set is allocated for each partition or piece, ensure that the data set naming convention in the template specification is unique enough. Use the &DSNUM variable, which resolves to a partition number or piece number at execution time.

**CONCURRENT**
Specifies that DFSMSdss concurrent copy is to make the full image copy. The image copy is recorded in the SYSIBM.SYSCOPY catalog table with ICTYPE=F and STYPE=C or STYPE=J.

If the SYSPRINT DD statement points to a data set, you must use a DSSPRINT DD statement.

When you specify SHRLEVEL(REFERENCE), an ICTYPE=Q record is placed into the SYSIBM.SYSCOPY catalog table after the object has been quiesced. If COPY fails, this record remains in SYSIBM.SYSCOPY. When COPY is successful, this ICTYPE=Q record is replaced with the ICTYPE=F record.

If the page size in the table space matches the control interval for the associated data set, you can use either the SHRLEVEL CHANGE option or the SHRLEVEL REFERENCE option with the CONCURRENT option. If the page size does not match the control interval, you must use the SHRLEVEL REFERENCE option for table spaces with a 8-KB, 16-KB, or 32-KB page size.

When you do not specify FILTERDDN, the DFSMSdss dump statement cannot include more than 255 data sets. When you request a concurrent copy on an object that exceeds this limitation, Db2 dynamically allocates a temporary filter data set for you.

**FILTERDDN**
Specifies the optional DD statement for the filter data set that COPY is to use.
with the CONCURRENT option. COPY uses this data set to automatically build a list of table spaces that are to be copied by DFSMSdss with one DFSMSdss DUMP statement.

You can use the FILTERDDN keyword to specify either a DD name or a TEMPLATE name specification from a previous TEMPLATE control statement. If utility processing detects that the specified name is both a DD name in the current job step and a TEMPLATE name, the utility uses the DD name.

If you specify FILTERDDN, the SYSCOPY records for all objects in the list have the same data set name.

*ddname* is the DD name.

**SHRLEVEL**

Indicates whether other programs can access or update the table space or index while COPY is running.

**REFERENCE**

Allows read-only access by other programs.

**CHANGE**

Allows other programs to change the table space or index space.

When you specify SHRLEVEL CHANGE, uncommitted data might be copied.

When SHRLEVEL CHANGE with FLASHCOPY CONSISTENT is specified, the COPY utility uses Db2 shadow processing to backout uncommitted work to make the FlashCopy image copy consistent without any availability outage to applications. Application updates are allowed throughout the creation of the FlashCopy image copy and the creation of the sequential image copies.

**Recommendation:** Except when creating FlashCopy image copies or traditional image copies with SHRLEVEL CHANGE and FLASHCOPY CONSISTENT specified, do not use image copies that are made with SHRLEVEL CHANGE when you run RECOVER TOCOPY.

SHRLEVEL CHANGE is not allowed for a table space that is defined as NOT LOGGED unless it is a LOB table space and its base table space has the LOGGED attribute.

SHRLEVEL CHANGE is not allowed when you use DFSMSdss concurrent copy for table spaces that have a page size that is greater than 4 KB and does not match the control interval size. If the page size in the table space matches the control interval size for the associated data set, you can use either the SHRLEVEL CHANGE option or the SHRLEVEL REFERENCE option.

If you are copying a list and you specify the SHRLEVEL CHANGE option, you can specify OPTIONS EVENT(ITEMERROR,SKIP) so that each object in the list is placed in UTRW status and the read claim class is held only while the object is being copied.

The read claim class is briefly obtained for each object during the UTILINIT phase to determine the object size if LIMIT is specified on the COPYDDN or RECOVERYDDN template. This applies only if OPTIONS EVENT(ITEMERROR,SKIP) is specified.
If you do not specify OPTIONS EVENT(ITEMERROR,SKIP), all of the objects in the list are placed in UTRW status and the read claim class is held on all objects for the entire duration of the COPY.

**SCOPE**

Indicates the scope of the copy for the specified objects.

**ALL**

Indicates that you want to copy all of the specified objects.

**PENDING**

Indicates that you want to copy only those objects in COPY-pending or informational COPY-pending status. When the DSNUM ALL option is specified for partitioned objects, and one or more of the partitions are in COPY-pending or informational COPY-pending status, a copy will be taken of the entire table space or index space.

For partitioned objects, if you only want the partitions in COPY-pending status or informational COPY-pending status to be copied, then a list of partitions should be specified. This is done by invoking COPY on a LISTDEF list built with the PARTLEVEL option. An output image copy data set will be created for each partition that is in COPY-pending or informational COPY-pending status.

**Related reference:**

Chapter 18, “LISTDEF,” on page 215

Chapter 34, “TEMPLATE,” on page 793

---

**Before running COPY**

Certain activities might be required before you run the COPY utility, depending on your situation.

Before running COPY, check that the table spaces and index spaces that you want to copy are not in any restricted states.

**Data sets that COPY uses**

The COPY utility uses a number of data sets during its operation.

The following table lists the data sets that COPY uses. The table lists the DD name that is used to identify the data set, a description of the data set, and an indication of whether it is required. Include statements in your JCL for each required data set and any optional data sets that you want to use.

**Table 18. Data sets that COPY uses**

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSIN</td>
<td>Input data set that contains the utility control statement.</td>
<td>Yes</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>Output data set for messages.</td>
<td>Yes</td>
</tr>
<tr>
<td>DSSPRINT</td>
<td>Output data set for messages when making concurrent copies or FlashCopy image copies.</td>
<td>No¹</td>
</tr>
</tbody>
</table>

¹ Optional required.
Table 18. Data sets that COPY uses (continued)

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter</td>
<td>A single data set that Db2 uses when you specify the FILTERDDN option in the utility control statement. This data set contains a list of VSAM data set names that Db2 builds, and is used during COPY when you specify the CONCURRENT and FILTERDDN options.</td>
<td>No$^2$</td>
</tr>
<tr>
<td>Sequential image copies</td>
<td>From one to four sequential output data sets that contain the resulting sequential format image copy data sets. Specify their DD names with the COPYDDN and RECOVERYDDN options of the utility control statement. The default is one copy to be written to the data set described by the SYSCOPY DD statement.</td>
<td>Yes</td>
</tr>
<tr>
<td>FlashCopy image copies</td>
<td>For table space or index space level copies, a VSAM data set for the output FlashCopy image copy of each partition or piece. For a partition level or piece level copy, a VSAM data set for the output FlashCopy image copy of the partition or piece.</td>
<td>No$^3$</td>
</tr>
</tbody>
</table>

Note:

1. Required if you specify CONCURRENT, FLASHCOPY YES, or FLASHCOPY CONSISTENT and the SYSPRINT DD statement points to a data set.
2. Required if you specify the FILTERDDN option.
3. Required if you specify either FLASHCOPY YES or FLASHCOPY CONSISTENT.

The following objects are named in the utility control statement and do not require DD statements in the JCL:

**Table space or index space**
Object that is to be copied. (If you want to copy only certain data sets in a table space, you must use the DSNUM option in the control statement.)

**Db2 catalog objects**
Objects in the catalog that COPY accesses. The utility records each copy in the Db2 catalog table SYSIBM.SYSCOPY.

**Output data set size**

Sequential image copies are written to sequential non-VSAM data sets.

FlashCopy image copies are written to VSAM data sets.

**Recommendation:** Use a template for the image copy data set by specifying a TEMPLATE statement without the SPACE keyword. When you omit this keyword, the utility calculates the appropriate size of the data set for you.

Alternatively, you can find the approximate size of the image copy data set for a table space, in bytes, by either executing COPY with the CHANGELIMIT REPORTONLY option, or using the following procedure:
1. Find the high-allocated page number, either from the NACTIVEF column of SYSIBM.SYSTABLESPACE after running the RUNSTATS utility, or from information in the VSAM catalog data set.

2. Multiply the high-allocated page number by the page size.

Filter data set size

Recommendation: Use a template for the filter data set by specifying a TEMPLATE statement without the SPACE keyword. When you omit this keyword, the utility calculates the appropriate size of the data set for you.

Alternatively, you can determine the approximate size of the filter data set size that is required, in bytes, by using the following formula, where \( n \) = the number of specified objects in the COPY control statement:

\[
(240 + (80 \times n))
\]

JCL parameters

For the output data sets of sequential format image copies, you can specify a block size by using the BLKSIZE parameter on the DD statement for the output data set. Valid block sizes are multiples of 4096 bytes. You can increase the buffer using the BUFNO parameter; for example, you might specify BUFNO=30, which creates 30 buffers.

For the output data sets of image copies created by invoking either the concurrent copy function or FlashCopy function of DFSMSdss, the required specifications for the output data sets might be different than those for sequential format image copies. For example, the BUFNO parameter does not apply to the output data sets for concurrent image copies.

Cataloging image copies

To catalog your image copy data sets, use the DISP=(MOD,CATLG,CATLG) parameter in the DD statement or TEMPLATE that is named by the COPYDDN option. After the image copy is taken, the DSVOLSER column of the row that is inserted into SYSIBM.SYSCOPY contains blanks.

FlashCopy image copy data sets are always cataloged. The DISP= parameter is not specified in the FlashCopy template. After the image copy is taken, unless uncommitted work is backed out of the image copy when FLASHCOPY CONSISTENT is specified, the DSVOLSER column of the row that is inserted into SYSIBM.SYSCOPY contains blanks. If uncommitted work is backed out of a FlashCopy image copy, the DSVOLSER column contains the Db2 checkpoint information for each member.

Duplicate image copy data sets are not allowed. If a cataloged data set is already recorded in SYSIBM.SYSCOPY with the same name as the new image copy data set, the COPY utility issues a message and does not make the copy.

When RECOVER locates the SYSCOPY entry, it uses the operating system catalog to allocate the required data set. If you have uncataloged the data set, the allocation fails. In that case, the recovery can still go forward; RECOVER searches for a previous image copy. But even if it finds one, RECOVER must use correspondingly more of the log during recovery.
**Recommendation:** Keep the ICF catalog consistent with the information about existing image copy data sets in the SYSIBM.SYSCOPY catalog table.

**Related concepts:**
- “Data sets that online utilities use” on page 16

**Concurrency and compatibility for COPY**

The COPY utility has certain concurrency and compatibility characteristics associated with it.

Db2 treats individual data and index partitions as distinct target objects. Utilities that operate on different partitions of the same table space or index space are compatible.

**Restricted states**

Do not copy a table space that is in any of the following states:
- CHECK-pending
- RECOVER-pending
- REFRESH-pending
- Logical error range
- Group buffer pool RECOVER-pending
- Stopped
- STOP-pending
- PRO restricted status

Do not copy an index space that is in any of the following states:
- CHECK-pending
- REBUILD-pending
- RECOVER-pending
- REFRESH-pending
- Logical error range
- Group buffer pool RECOVER-pending
- Stopped
- STOP-pending

If a table space is in COPY-pending status, or a table space or index is in informational COPY-pending status, you can reset the status only by taking a full image copy of the entire table space, all partitions of the partitioned table space, or the index space. When you make an image copy of a partition, the COPY-pending status of the partition is reset. If a nonpartitioned table space is in COPY-pending status, you can reset the status only by taking a full image copy of the entire table space, and not of each data set.

**Claims and drains**

The following table shows which claim classes COPY claims and drains and any restrictive status that the utility sets on the target object.

*Table 19. Claim classes of COPY operations*

<table>
<thead>
<tr>
<th>Target</th>
<th>SHRLEVEL REFERENCE</th>
<th>SHRLEVEL CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table space, index space, or partition</td>
<td>DW UTRO</td>
<td>CR UTRW³</td>
</tr>
</tbody>
</table>
Table 19. Claim classes of COPY operations (continued)

<table>
<thead>
<tr>
<th>Target</th>
<th>SHRLEVEL REFERENCE</th>
<th>SHRLEVEL CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Legend:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• DW - Drain the write claim class - concurrent access for SQL readers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• CR - Claim the read claim class</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• UTRO - Utility restrictive state, read-only access allowed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• UTRW - Utility restrictive state, read-write access allowed</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. If the target object is a segmented table space, SHRLEVEL CHANGE does not allow you to concurrently execute an SQL DELETE without the WHERE clause.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

COPY does not set a utility restrictive state if the target object is DSNDB01.SYSUTILX.

Compatibility

The following table documents which utilities can run concurrently with COPY on the same target object. The target object can be a table space, an index space, or a partition of a table space or index space. If compatibility depends on particular options of a utility, that information is also documented in the table.

Table 20. Compatibility of COPY with other utilities

<table>
<thead>
<tr>
<th>Action</th>
<th>COPY INDEXSPACE SHRLEVEL REFERENCE</th>
<th>COPY INDEXSPACE SHRLEVEL CHANGE</th>
<th>COPY TABLESPACE SHRLEVEL REFERENCE</th>
<th>COPY TABLESPACE SHRLEVEL CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACKUP SYSTEM</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CHECK DATA</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CHECK INDEX</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CHECK LOB</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COPY INDEXSPACE</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COPY TABLESPACE</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>COPYTOCOPY</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DIAGNOSE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LOAD</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MERGECOPY</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MODIFY</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>QUIESCE</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>REBUILD INDEX</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RECOVER INDEX</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RECOVER TABLESPACE</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REORG INDEX</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>REORG TABLESPACE SHRLEVEL CHANGE</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes²</td>
</tr>
<tr>
<td>REORG TABLESPACE UNLOAD CONTINUE or PAUSE</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Table 20. Compatibility of COPY with other utilities (continued)

<table>
<thead>
<tr>
<th>Action</th>
<th>COPY INDEXSPACE SHRLEVEL CHANGE</th>
<th>COPY INDEXSPACE SHRLEVEL CHANGE</th>
<th>COPY TABLESPACE SHRLEVEL CHANGE</th>
<th>COPY TABLESPACE SHRLEVEL CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>REORG TABLESPACE UNLOAD ONLY or EXTERNAL</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>REPAIR LOCATE by KEY, RID, or PAGE DUMP or VERIFY</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>REPAIR LOCATE by KEY or RID DELETE or REPLACE</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REPAIR LOCATE INDEX PAGE REPLACE</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>REPAIR LOCATE TABLESPACE PAGE REPLACE</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REPORT</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RESTORE SYSTEM</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RUNSTATS INDEX</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RUNSTATS TABLESPACE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>STOSPACE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UNLOAD(^1)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Note:

1. If CONCURRENT option is used, contention might be encountered when other utilities are run on the same object at the same time.

2. REORG TABLESPACE SHRLEVEL CHANGE and COPY SHRLEVEL CHANGE are compatible and can run concurrently except during the period when exclusive control is needed to drain claimers of a target table space.

Restriction:

- COPY with the FLASHCOPY CONSISTENT option is not compatible with REORG.
- If REORG has drained the claimers of a table space or table space partition and a COPY utility is submitted to access the same object, the COPY utility terminates with a message that it is not compatible.
- If COPY and REORG are accessing the same table space or table space partitions, REORG cannot drain claimers until COPY completes. The REORG DRAIN options determine the actions taken.
- If COPY and REORG are accessing the same table space or table space partitions and COPY abends, restart of the COPY is not allowed if REORG completes.

To run on DSNDB01.SYSUTILX, COPY must be the only utility in the job step. Also, if SHRLEVEL REFERENCE is specified, the COPY job of DSNDB01.SYSUTILX must be the only utility running in the Sysplex.

COPY on SYSUTILX is an “exclusive” job; such a job can interrupt another job between job steps, possibly causing the interrupted job to time out.

Related tasks:

"Monitoring utilities” on page 39

Related reference:
Full image copies

You can make full image copies of a variety of data objects. Data objects include table spaces, table space partitions, data sets of nonpartitioned table spaces, index spaces, and index space partitions.

The following statement specifies that the COPY utility is to make a full image copy of the DSN8S11E table space in database DSN8D11A:

COPY TABLESPACE DSN8D11A.DSN8S11E

The COPY utility writes pages from the table space or index space to the output data sets. The JCL for the utility job must include DD statements or have a template specification for the data sets. If the object consists of multiple data sets and all are copied in one run, the copies reside in one physical sequential output data set.

For sequential image copies, if the object consists of multiple data sets and all are copied in one run, the copies reside in one physical sequential output data set. For FlashCopy image copies, if the object consists of multiple data sets and all are copied in one run, there is a FlashCopy image copy data set for each data set.

Image copies should be made either by entire page set or by partition, but not by both.

Recommendations:

- Take a full image copy after any of the following operations:
  - CREATE or LOAD operations for a new object that is populated.
  - REORG operation for an existing object.
  - LOAD RESUME of an existing object.
  - LOGGED operation of a table space.
- Copy the indexes over a table space whenever a full copy of the table space is taken. More frequent index copies decrease the number of log records that need to be applied during recovery. At a minimum, you should copy an index when it is placed in informational COPY-pending (ICOPY) status.

If you create an inline copy during LOAD or REORG, you do not need to execute a separate COPY job for the table space. If you do not create an inline copy, and if the LOG option is NO, the COPY-pending status is set for the table space. You must then make a full image copy for any subsequent recovery of the data. An incremental image copy is not allowed in this case. If the LOG option is YES, the COPY-pending status is not set. However, your next image copy must be a full image copy. Again, an incremental image copy is not allowed.

The COPY utility automatically takes a full image copy of a table space if you attempt to take an incremental image copy when it is not allowed.

If a table space changes after an image copy is taken and before the table space is altered from NOT LOGGED to LOGGED, the table space is marked COPY-pending, and a full image copy must be taken.

The catalog table SYSIBM.SYSCOPY and the directory tables SYSIBM.SYSUTILX and SYSIBM.SYSLGRNX record information from the COPY utility.
table spaces for those tables can lock out separate COPY jobs that are running simultaneously; therefore, defer copying the table spaces for SYSIBM.SYSCOPY, SYSIBM.SYSUTILX, and SYSIBM.SYSLGRNX until the other copy jobs have completed. However, if you must copy other objects while another COPY job processes the catalog or directory, specify SHRLEVEL(CHANGE) for copying the catalog and directory table spaces. Beginning in DB2 10, the COPY control statements in the DSNTIJIC job specify SHRLEVEL(CHANGE).

**Related concepts:**
“Copying catalog and directory objects” on page 165

**Related reference:**
Appendix B, “Advisory or restrictive states,” on page 1037

---

### Incremental image copies

An incremental image copy is a copy of the pages that have been changed since the last full or incremental image copy.

You cannot take an incremental image copy of an index space.

You can make an incremental image copy of a table space if the following conditions are true:
- A full image copy of the table space exists.
- The COPY-pending status is not on for that table space.
- The last copy was taken without the CONCURRENT option.
- The previous copy was not made by using FlashCopy. This applies even if sequential copies were made from the FlashCopy image copy.

**Copy by partition or data set**

You can make an incremental image copy by partition or data set (specified by DSNUM) in the following situations:
- A full image copy of the table space exists.
- A full image copy of the same partition or data set exists and the COPY-pending status is not on for the table space or partition.

In addition, the full image copy must have been made after the most recent use of CREATE, REORG or LOAD, or it must be an inline copy that was made during the most recent use of LOAD or REORG.

**Sample control statement**

To specify an incremental image copy, use FULL NO on the COPY statement, as in the following example:

```
COPY TABLESPACE DSN8D11A.DSN8S11E FULL NO SHRLEVEL CHANGE
```

**Performance advantage**

An incremental image copy generally does not require a complete scan of the table space, with two exceptions:
- The table space is defined with the TRACKMOD NO option.
- You are taking the first copy after you altered a table space to TRACKMOD YES.

Space maps in each table space indicate, for each page, regardless of whether it has changed since the last image copy. Therefore, making an incremental copy can be
significantly faster than making a full copy if the table space is defined with the TRACKMOD YES option. Incremental image copies of a table space that is defined with TRACKMOD NO still saves space, although the performance advantage is smaller.

Restriction: You cannot make incremental copies of DSNDB01.DBD01, DSNDB01.SYSDBDXA, DSNDB01.SYSUTILX, or DSNDB06.SYSTSCPY in the catalog. For those objects, COPY always makes a full image copy and places the SYSCOPY record in the log.

Multiple image copies

In a single COPY job, you can create up to five exact copies of various data objects. Data objects include table spaces, table space partitions, data sets of a nonpartitioned table space, index spaces, and index space partitions.

You can make two sequential copies for use on the local Db2 system (installed with the option LOCALSITE), and two more for offsite recovery (on any system that is installed with the option RECOVERSITE). You can also make a fifth FlashCopy image copy for use on the local Db2 system. All copies are identical, and all are produced at the same time from one invocation of COPY.

Alternatively you can use COPYTOCOPY to create the needed image copies. COPYTOCOPY can be used to create sequential image copies from a FlashCopy image copy.

The ICBACKUP column in SYSIBM.SYSCOPY specifies whether the image copy data set is for the local or recovery system, and whether the image copy data set is for the primary copied data set or for the backup copied data set. The ICUNIT column in SYSIBM.SYSCOPY specifies whether the image copy data set is on tape or disk.

Remote-site recovery

For remote site recovery, Db2 assumes that the system and application libraries and the Db2 catalog and directory are identical at the local site and recovery site. You can regularly transport copies of archive logs and database data sets to a safe location to keep current data for remote-site recovery current. This information can be kept on tape until needed.

Naming the data sets for the copies

The COPYDDN option of COPY names the output data sets that receive copies for local use. The RECOVERYDDN option names the output data sets that receive copies that are intended for remote-site recovery. The options have the following formats:

COPYDDN (ddname1,ddname2)    RECOVERYDDN (ddname3,ddname4)

The DD names for the primary output data sets are ddname1 and ddname3. The ddnames for the backup output data sets are ddname2 and ddname4.

Sample control statement

The following statement makes four full image copies of the table space DSN8S11E in database DSN8D11A. The statement uses LOCALDD1 and LOCALDD2 as DD
names for the primary and backup copies that are used on the local system and
RECOVDD1 and RECOVDD2 as DD names for the primary and backup copies for
remote-site recovery:

COPY TABLESPACE DSN8D11A.DSN8S11E
COPYDDN (LOCALDD1,LOCALDD2)
RECOVERYDDN (RECOVDD1,RECOVDD2)

You do not need to make copies for local use and for remote-site recovery at the
same time. COPY allows you to use either the COPYDDN or the RECOVERYDDN
option without the other. If you make copies for local use more often than copies
for remote-site recovery, a remote-site recovery could be performed with an older
copy, and more of the log, than a local recovery; hence, the recovery would take
longer. However, in your plans for remote-site recovery, that difference might be
acceptable. You can also use MERGECOPY RECOVERYDDN to create recovery-site
full image copies, and merge local incremental copies into new recovery-site full
copies.

Conditions for making multiple incremental image copies

Db2 cannot make incremental image copies if any of the following conditions is
true:

- The incremental image copy is requested only for a site other than the current
  site (the local site from which the request is made).
- Incremental image copies are requested for both sites, but the most recent full
  image copy was made for only one site.
- Incremental image copies are requested for both sites and the most recent full
  image copies were made for both sites, but between the most recent full image
  copy and current request, incremental image copies were made for the current
  site only.

If you attempt to make incremental image copies under any of these conditions,
COPY terminates with return code 8, does not take the image copy or update the
SYSIBM.SYSCOPY table, and issues the following message:

DSNU404I csect-name LOCAL SITE AND RECOVERY SITE INCREMENTAL
IMAGE COPIES ARE NOT SYNCHRONIZED

To proceed, and still keep the two sets of data synchronized, take another full
image copy of the table space for both sites, or change your request to make an
incremental image copy only for the site at which you are working.

Db2 cannot make an incremental image copy if the object that is being copied is an
index or index space.

Maintaining copy consistency

Make full image copies for both the local and recovery sites:
- If a table space is in COPY-pending status
- After a LOAD or REORG procedure that did not create an inline copy
- If an index is in the informational COPY-pending status
- If a table space is in informational COPY-pending status

This action helps to ensure correct recovery for both local and recovery sites. If the
requested full image copy is for one site only, but the history shows that copies
were made previously for both sites, COPY continues to process the image copy
and issues the following warning message:
The COPY-pending status of a table space is not changed for the other site when you make multiple image copies at the current site for that other site. For example, if a table space is in COPY-pending status at the current site, and you make copies from there for the other site only, the COPY-pending status is still on when you bring up the system at that other site.

Related reference:
Chapter 15, “COPYTOCOPY,” on page 185

FlashCopy image copies

FlashCopy image copies can reduce both the time that data is unavailable during the copy operation and the time that is required for backup and recovery operations. Certain Db2 utilities can create these copies by invoking the FlashCopy function that is provided by z/OS DFSMS and the IBM TotalStorage Enterprise Storage Server® (ESS) storage subsystems.

DFSMSdss creates the FlashCopy image copies on cataloged VSAM data sets. A separate VSAM data set is created for each partition or piece of the object that is being copied.

The other image copies that are created by Db2 utilities (those copies that are not FlashCopy image copies) are written to non-VSAM sequential format data sets.

Utilities that support FlashCopy

The following Db2 for z/OS utilities can use FlashCopy to create image copies:
- COPY
- LOAD
- REBUILD INDEX
- REORG INDEX
- REORG TABLESPACE

When creating a FlashCopy image copy, the COPY and LOAD utilities with SHRLEVEL CHANGE can include additional phases of execution, depending on the options that are specified in the utility control statement. The additional execution phases include:
- SEQCOPY
- LOGAPPLY
- LOGCSR
- LOGUNDO

The following utilities accept the VSAM data sets that are produced by FlashCopy as input:
- COPYTOCOPY
- DSN1COMP
- DSN1COPY
- DSN1PRNT
- RECOVER
Using a FlashCopy image copy for porting data: Use the FlashCopy image copy as input to the DSN1COPY utility. If your data is versioned (with an immediate ALTER statement), run the REORG utility before creating the FlashCopy image copy.

Using FlashCopy image copies with the UNLOAD utility: The UNLOAD utility cannot operate directly on a FlashCopy image copy. To unload data from a FlashCopy image copy, first use the COPYTOCOPY utility to create a sequential format image copy from the FlashCopy image copy. Then, run the UNLOAD job on the sequential copy. In certain cases, individual rows might not be properly unloaded, in which case a warning message is issued. This situation might occur in two circumstances, mentioned below, but you can take actions to avoid the problem.

- If your data is versioned (with an immediate ALTER statement), run REORG on your data before creating the FlashCopy image copy.
- If your data has been compressed (by a compression during an insert operation), run the REORG on your data before creating the FlashCopy image copy. REORG is required only once after the compression during the insert operation occurred.

Requesting FlashCopy image copies

For utilities that support the creation of FlashCopy image copies, you can specify that you want these copies by using Db2 subsystem parameters, utility control statement options, or both.

To request a FlashCopy image copy for a particular utility job, specify FLASHCOPY YES in the utility control statement.

To request that a utility create FlashCopy image copies as the default behavior, specify YES for one or more of the following subsystem parameters:

- FLASHCOPY_COPY subsystem parameter
- FLASHCOPY_LOAD subsystem parameter
- FLASHCOPY_REORG_TS subsystem parameter
- FLASHCOPY_REBUILD_INDEX subsystem parameter
- FLASHCOPY_REORG_INDEX subsystem parameter

When these subsystem parameters are set to YES, you do not need to specify the FLASHCOPY option in the utility control statement. However, any FLASHCOPY value that you specify in a utility control statement overrides the value of the subsystem parameter.

When creating a FlashCopy image copy, the following utilities can also create one to four additional sequential format image copies in a single execution:

- COPY
- LOAD with the REPLACE option specified
- REORG TABLESPACE

The COPYTOCOPY utility can create sequential format image copies by using an existing FlashCopy image copy as input.

Recommendation: To provide a recovery base for media failures, create one or more additional sequential format image copies when you create a FlashCopy image copy.
A return code of 8 or greater from DFSMSdss results in termination of the utility FlashCopy process, even if OPTIONS EVENT(ITEMERROR,SKIP) was specified.

Related information:

"Subsystem parameters for refining DFSMSdss COPY operation with utilities” on page 40
Directory of subsystem parameters, panel fields, and application default values (Db2 Installation and Migration)

Data Set Level FlashCopy with disk mirroring

By default, FlashCopy operations are allowed to a primary volume in a z Global Mirror (Extended Remote Copy or XRC) relationship when the z/OS DFSMSdss support for RPFC for XRC is installed and enabled. If you want to change this default behavior, change the value of the FLASHCOPY_XRCP subsystem parameter.

Related information:

FLASHCOPY XRCP field (FLASHCOPY_XRCP subsystem parameter) (Db2 Installation Guide)

Image copy consistency with FlashCopy

When you create FlashCopy image copies with COPY or LOAD with the SHRLEVEL CHANGE option, you can specify that you want that image copy to be consistent for recovery purposes. To ensure this consistency, specify the FLASHCOPY CONSISTENT option.

Restriction: You cannot specify CONSISTENT when copying objects that are defined with the NOT LOGGED attribute.

When you specify FLASHCOPY CONSISTENT, the utility checks the logs for changes to the copied data that were uncommitted at the time that the image copy was created. Any uncommitted data that is identified in the logs is backed out of the image copy before the utility terminates. Therefore, the process of creating an consistent FlashCopy image copy uses more system resources and takes longer than creating a FlashCopy image copy without specifying FLASHCOPY CONSISTENT.

The utilities use Db2 shadow processing to make the FlashCopy image copy consistent. The FlashCopy image copy data set is used as the shadow. The naming convention that is used in this case is different than the naming convention that is used by other utilities for Db2 shadow processing.

Operational restrictions for FlashCopy

Only one utility can create a FlashCopy image copy of an object at a time. For example, suppose that the COPY utility with the SHRLEVEL CHANGE and FLASHCOPY options is running on an object. Then, the LOAD utility with the SHRLEVEL CHANGE and FLASHCOPY options starts running on the same object. The LOAD utility receives message DSNU180I with a return code of 8 to indicate that the LOAD utility is not compatible with the COPY utility.
A utility cannot create FlashCopy image copies in the following situations: (In these situations, the term data set refers to a Db2 table space or index space or a FlashCopy image copy.)

- FlashCopy Version 2 disk volumes are not available.
- The source data set is already the target of a FlashCopy relationship.
- The target data set is already the source of a FlashCopy relationship.
- The source data set is already participating in the maximum number of FlashCopy relationships.
- The CISIZE, CASIZE, physical record size, or physical block size of the target data set is different from that of the source data set. The CASIZE of the target data set can be different from the source data set if the source data set is less than one cylinder.
- The source data set and the target data set are not both fully contained on the same physical control unit (controller).

**Recommendation:** Use the storage class attribute ACCESSIBILITY=REQUIRED or ACCESSIBILITY=PREFERRED for both the source data set and the target data set. If the storage class that is associated with a data set has this attribute, DFSMS attempts to select volumes such that the data set is contained on volumes within a single physical control unit.

- Either the source data set or the target data set is not SMS-managed.

For more information about FlashCopy restrictions, see Moving Data Sets with FlashCopy (z/OS DFSMSdss Storage Administration).

If FlashCopy cannot be used, the utility creates sequential format image copies of the object. This process can take longer than the expected execution time for creating FlashCopy image copies.

**SYSCOPY records for FlashCopy image copies**

For each FlashCopy image copy, Db2 creates one or more records in the SYSIBM.SYSCOPY table. These SYSCOPY records can differ slightly from the records for sequential format image copies depending on whether the object that is being copied is partitioned and the number of partitions or object pieces that are being copied.

For a FlashCopy image copy of a single partition or piece of an object, one SYSCOPY record is created for the partition or piece.

For a FlashCopy image copy of a table space or index space that contains more than one partition or piece, Db2 creates a separate SYSCOPY record for each of the following items:

- The table space or index space
- Each partition or piece in the table space or index space

In the DSNUM column of the SYSCOPY record, Db2 assigns a data set number to the table space or index space and to each partition or piece. The data set numbers start at 0 for the table space or index space and are incremented by 1 for each partition or piece.

For example, if a table space has two partitions, a FlashCopy image copy of the table space creates three SYSCOPY records as follows:
• A SYSCOPY record for the tables space with DSNUM=0
• A SYSCOPY record for the first partition with DSNUM=1
• A SYSCOPY record for the second partition with DSNUM=2

For a FlashCopy image copy of a table space or index space that contains only one partition or piece, only one SYSCOPY record is created with DSNUM=0.

For FlashCopy image copies that were created without the FLASHCOPY CONSISTENT option, the START_RBA value corresponds to the RBA or LRSN when the object's pages were last externalized to disk.

For FlashCopy image copies that were created with the FLASHCOPY CONSISTENT option, the START_RBA value depends on whether active units of work existed. If active units of work existed, the START_RBA value corresponds to the beginning RBA or LRSN of the oldest uncommitted unit of work that was backed out. If no active units of work existed, the START_RBA value corresponds to the RBA or LRSN when the object's pages were last externalized to disk.

The implication of the START_RBA value for FlashCopy image copies is that a recovery from a FlashCopy image copy likely requires more log processing.

In the SYSCOPY section of the output from REPORT RECOVER, the SYSCOPY records are presented in ascending START_RBA order and not in timestamp order. Thus, the SYSCOPY records for FlashCopy image copies might be shown in the REPORT RECOVER out of chronological order as compared to other SYSCOPY records.

Related tasks:
- “Recovering with FlashCopy image copies” on page 481

Related reference:
- “Syntax and options of the COPY control statement” on page 135
- “Syntax and options of the LOAD control statement” on page 242
- “Syntax and options of the REBUILD INDEX control statement” on page 428
- “Syntax and options of the REORG INDEX control statement” on page 516
- “Syntax and options of the REORG TABLESPACE control statement” on page 556

Copies of lists of objects

Within a single COPY control statement, the COPY utility allows you to process a list that contains any of the following objects: table space, table space partition, data set of a nonpartitioned table space, index space, and index space partition.

Specifying objects in a list is useful for copying a complete set of referentially related table spaces before running QUIESCE. Consider the following information when taking an image copy for a list of objects:

• Db2 copies table spaces and index spaces in the list one at a time, in the specified order, unless you invoke parallelism by specifying the PARALLEL keyword.
• Each table space in the list with a CHANGELIMIT specification has a REPORT phase, so the phase switches between REPORT and COPY while processing the list.
• If processing completes successfully, any COPY-pending or informational COPY-pending status on the table spaces and informational COPY-pending status on the indexes are reset.
If you use COPY with the SHRLEVEL(REFERENCE) option:
- Db2 drains the write claim class on each table space and index in the UTILINIT phase, which is held for the duration of utility processing.
- Utility processing inserts SYSCOPY rows for all of the objects in the list at the same time, after all of the objects have been copied.
- All objects in the list have identical RBA or LRSN values for the START_RBA column for the SYSCOPY rows: the START_RBA is set to the current LRSN at the end of the COPY phase.

If you use COPY with the SHRLEVEL(CHANGE) option:
- If you specify OPTIONS EVENT(ITEMERROR,SKIP), each object in the list is placed in UTRW status and the read claim class is held only while the object is being copied. If you do not specify OPTIONS EVENT(ITEMERROR,SKIP), all of the objects in the list are placed in UTRW status and the read claim class is held on all objects for the entire duration of the COPY.
- Utility processing inserts a SYSCOPY row for each object in the list when the copy of each object is complete.
- Objects in the list have different LRSN values for the START_RBA column for the SYSCOPY rows; the START_RBA value is set to the current RBA or LRSN at the start of copy processing for that object.

When you specify the PARALLEL keyword, Db2 supports parallelism for image copies on disk or tape devices. You can control the number of tape devices to allocate for the copy function by using TAPEUNITS with the PARALLEL keyword. If you use JCL statements to define tape devices, the JCL controls the allocation of the devices.

When you explicitly specify objects with the PARALLEL keyword, the objects are not necessarily processed in the specified order. Objects that are to be written to tape and whose file sequence numbers have been specified in the JCL are processed in the specified order. If templates are used, you cannot specify file sequence numbers. In the absence of overriding JCL specifications, Db2 determines the placement and, thus, the order of processing for such objects. When only templates are used, objects are processed according to their size, with the largest objects processed first.

Both the PARALLEL and TAPEUNITS keywords act as constraints on the processing of the COPY utility. The PARALLEL keyword constrains the amount of parallelism by restricting the maximum number of objects that can be processed simultaneously. The TAPEUNITS keyword constrains the number of tape drives that can be dynamically allocated for the COPY command. The TAPEUNITS keyword can constrain the amount of parallelism if an object requires a number of tapes such that the number of remaining tapes is insufficient to service another object.

To calculate the number of threads that you need when you specify the PARALLEL keyword, use the formula \((n * 2 + 1)\), where \(n\) is the number of objects that are to be processed in parallel, regardless of the total number of objects in the list. If you do not use the PARALLEL keyword, \(n\) is 1 and COPY uses three threads for a single-object COPY job.

COPY SCOPE PENDING indicates that you want to copy only those objects in COPY-pending or informational COPY-pending status. When the DSNUM ALL
option is specified for partitioned objects, and one or more of the partitions are in COPY-pending or informational COPY-pending status, a copy will be taken of the entire table space or index space.

For partitioned objects, if you only want the partitions in COPY-pending status or informational COPY-pending status to be copied, then a list of partitions should be specified. It is recommended that you do this by invoking COPY on a LISTDEF list built with the PARTLEVEL option. An output image copy data set will be created for each partition that is in COPY-pending or informational COPY-pending status.

The LIMIT option on the TEMPLATE statement allows you to switch templates for output copy data sets. Template switching is most commonly needed to direct small data sets to DASD and large data sets to TAPE. This allows you to switch to templates that differ in the UNIT, DSNs, or HSM classes.

The following table spaces cannot be included in a list of table spaces. You must specify each one as a single object:
- DSNDB01.SYSUTILX
- DSNDB06.SYSTSCPY
- DSNDB01.SYSLGRNX
- DSNDB01.SYSDBDXA

The only exceptions to this restriction are the indexes over these table spaces that were defined with the COPY YES attribute. You can specify such indexes along with the appropriate table space.

Related reference:
Chapter 34, “TEMPLATE,” on page 793

Using more than one COPY statement

You can use more than one control statement for COPY in one Db2 utility job step.

About this task

After each COPY statement executes successfully:
- A row that refers to each image copy is recorded in the SYSIBM.SYSCOPY table.
- The image copy data sets are valid and available for RECOVER, MERGECOPY, COPYTOCOPY, and UNLOAD.

If a job step that contains more than one COPY statement abends, do not use TERM UTILITY. Restart the job from the last commit point by using RESTART instead. Terminating COPY by using TERM UTILITY in this case creates inconsistencies between the ICF catalog and Db2 catalogs.

Copying partitions or data sets simultaneously

To potentially improve the performance of the COPY utility, copy partitions or data sets at the same time.

Procedure

To copy partitions or data sets simultaneously:
- If the table space is partitioned, take one of the following actions:
  - Specify the PARALLEL option in the COPY utility control statement to copy partitions in the same COPY execution in parallel.
Copies of partition-by-growth table spaces

An image copy at the table space level with SHRLEVEL(CHANGE) will not contain any new partitions that are added by SQL INSERT operations after the image copy began. The newly added partitions can be recovered by using the Db2 logs.

When you make an image copy of a partition-by-growth table space, a partition might be empty as a result of REORG, SQL delete operations, or recovery to a prior point in time. The empty partition has a header page and space map pages or system pages. The COPY utility still copies the empty partition.

Copies of XML table spaces

The COPY utility supports full and incremental image copies for XML table spaces. The COPY utility options SHRLEVEL REFERENCE, SHRLEVEL CHANGE, CONCURRENT, and FLASHCOPY are also supported for XML table spaces.

Unless either the CONCURRENT option or the FLASHCOPY option is specified, COPY does not copy empty or unformatted data pages of an XML table space.

If you copy a LOB table space that has a base table space with the NOT LOGGED attribute, copy the base table space and the LOB table space together so that a RECOVER TOLOGPOINT of the entire set results in consistent data across the base table space and all of the associated LOB table spaces.

Note: RECOVER TOLASTCOPY is not allowed on a list of objects. Instead, use RECOVER TOLOGPOINT, where the TOLOGPOINT is the common RBA or LRSN associated with the SHRLEVEL REFERENCE image copies.

To copy an XML table space with a base table space that has the NOT LOGGED attribute, all associated XML table spaces must also have the NOT LOGGED attribute. The XML table space acquires this NOT LOGGED attribute by being linked to the logging attribute of its associated base table space. You cannot independently alter the logging attribute of an XML table space.

If the LOG column of the SYSIBM.SYSTABLESPACE record for an XML table space has the value of “X”, the logging attributes of the XML table space and its base table space are linked, and that the logging attribute of both table spaces is NOT LOGGED. To break the link, alter the logging attribute of the base table space back to LOGGED, and the logging attribute of both table spaces are changed back to LOGGED.
Copying catalog and directory objects

Use the DSNTIJIC installation job to create backup copies of catalog and directory objects.

If you are migrating from a Version 9 or earlier Db2 for z/OS system, and your Version 10 or later Db2 for z/OS system is in conversion mode, use the DSNTIJIC job that is produced by running the installation CLIST in MIGRATE mode.

Depending on the migration mode in which the Db2 for z/OS system is running, the COPY utility skips new or obsolete catalog and directory objects during processing and issues message DSNU1530I with RC0 for each skipped object. For example, in conversion mode, the COPY utility skips catalog and directory objects that are new for the version to which you are migrating. In new function mode, the COPY utility skips catalog and directory objects that are obsolete in the version to which you are migrating. When LISTDEF is used for catalog and directory objects, the COPY utility issues message DSNU054I for obsolete catalog or directory objects. If OPTIONS EVENTITEMERROR,SKIP is specified, the object is skipped, and RC 4 is issued. Otherwise, the COPY utility terminates with RC 8.

If the output image copy has a hardcoded DD statement to a tape device, the COPY utility opens and closes the data set to write a tape mark. This ensures that subsequent image copies stacked to the same tape volume can be written.

Specifying OPTIONS EVENTITEMERROR,SKIP or OPTIONS EVENTITEMERROR,HALT does not impact the skipping of new or obsolete objects.

The catalog table SYSIBM.SYSCOPY and the directory tables SYSIBM.SYSUTILX and SYSIBM.SYSLGRNX record information from the COPY utility. Copying the table spaces for those tables can lock out separate COPY jobs that are running simultaneously; therefore, defer copying the table spaces for SYSIBM.SYSCOPY, SYSIBM.SYSUTILX, and SYSIBM.SYSLGRNX until the other copy jobs have completed. However, if you must copy other objects while another COPY job processes the catalog or directory, specify SHRLEVEL(CHANGE) for copying the catalog and directory table spaces. Beginning in DB2 10, the COPY control statements in the DSNTIJIC job specify SHRLEVEL(CHANGE).

Related information:

DSNU1530I (Db2 Messages)

Make copies of XML schema repository objects

Although the XML schema repository is not part of the Db2 catalog, you need to create backup copies of XML schema repository table spaces when you back up catalog objects.

To determine which table spaces are part of the XML schema repository, see job DSNTESR.

The table spaces for which you need to make image copies are in database DSNXSR.
Copies of Indexes

If you copy a COPY YES index of a table space that has the NOT LOGGED attribute, copy the indexes and table spaces together. Copying the indexes and table spaces together ensures that the indexes and the table space have the same recoverable point.

An index should be image copied after an ALTER INDEX REGENERATE. You should copy the index after it has been rebuilt for these types of ALTER statements:
- alter to padded
- alter to not padded
- alter add of a key column
- alter of a numeric data type key column

Any new partitions added by SQL INSERT are not contained in the image copy, but the newly added partitions are recoverable by the Db2 logs.

When the index has the COMPRESS YES attribute, concurrent copies of indexes and FlashCopy image copies of indexes are compressed because DFSMSdss is invoked to copy the VSAM linear data sets (LDS) for the index. Image copies of indexes are not compressed because the index pages are copied from the Db2 buffer pool. When image copies are taken without the concurrent option, you can choose to compress the image copies by using access method compression via DFSMS or by using IDRC if the image copies reside on tape.

Using DFSMSdss concurrent copy

You might be able to gain improved availability by using the concurrent copy function of the DFSMSdss component of the Data Facility Storage Management Subsystem (DFSMS). To complete recovery, you can subsequently run the Db2 RECOVER utility to restore those image copies and apply the necessary log records to them.

The CONCURRENT option of COPY invokes DFSMSdss concurrent copy. The COPY utility records the resulting DFSMSdss concurrent copies in the catalog table SYSIBM.SYSCOPY with ICTYPE=F and STYPE=C or STYPE=J. STYPE=C indicates that the concurrent copy was taken of the 'I' instance of the table space (which means that the instance qualifier in the name of the corresponding data set begins with the letter 'I'). STYPE=J indicates that the concurrent copy was taken of the 'J' instance of the table space (which means that the instance qualifier in the name of the corresponding data set begins with the letter 'J').

To obtain a consistent offline backup copy outside of Db2:
1. Start the Db2 objects that are being backed up for read-only access by issuing the following command:
   ```bash
   START DATABASE(database-name) SPACENAM(tablespace-name) ACCESS(RO)
   ```
   Allowing read-only access is necessary to ensure that no updates to data occur during this procedure.
2. Run QUIESCE with the WRITE(YES) option to quiesce all Db2 objects that are being backed up.
3. Back up the Db2 data sets after the QUIESCE utility completes successfully.
4. Issue the following command to allow transactions to access the data:

```bash
-START DATABASE(database-name) SPACENAM(tablename)
```

If you use the CONCURRENT option:

- You must supply either a COPYDDN DD name, a RECOVERYDDN DD name, or both. Note that the required JCL parameter specifications for the output data sets for the CONCURRENT option might differ from the JCL specifications required for sequential format data sets. For example, do not specify the BUFNO parameter for the output data sets when specifying the CONCURRENT option.
- You can set the disposition to DISP=(MOD,CATLG,CATLG) if you specify the new data set for the image copy on a scratch volume (a specific volume serial number is not specified). You must set the disposition to DISP=(NEW,CATLG,CATLG) if you specify a specific volume serial number for the new image copy data set.
- If you are restarting COPY, specify DISP=(MOD,CATLG,CATLG) or DISP=(NEW,CATLG,CATLG) for the COPYDDN and RECOVERYDDN data sets. The DFSMSdss DUMP command does not support appending to an existing data set. Therefore, the COPY utility converts any DISP=MOD data sets to DISP=OLD before invoking DFSMSdss.
- If the SYSPRINT DD statement points to a data set, you must use a DSSPRINT DD statement.
- If the page size in the table space matches the control interval for the associated data set, you can use either the SHRLEVEL CHANGE option or the SHRLEVEL REFERENCE option. If the page size does not match the control interval, you must use the SHRLEVEL REFERENCE option for table spaces with a 8-KB, 16-KB, or 32-KB page size.

**Restrictions on using DFSMSdss concurrent copy**

You cannot use a copy that is made with DFSMSdss concurrent copy with the PAGE or ERROR RANGE options of the RECOVER utility. If you specify PAGE or ERROR RANGE, RECOVER bypasses any concurrent copy records when searching the SYSIBM.SYSCOPY table for a recovery point.

You can use the CONCURRENT option with SHRLEVEL CHANGE on a table space if the page size in the table space matches the control interval for the associated data set.

Also, you cannot run the following Db2 stand-alone utilities on copies that are made by DFSMSdss concurrent copy:

```bash
DSN1COMP
DSN1COPY
DSN1PRNT
```

You cannot execute the CONCURRENT option from the DB2I Utilities panel or from the DSNU TSO CLIST command.

**Requirements for using DFSMSdss concurrent copy**

DFSMSdss concurrent copy is enabled by specific hardware. Contact IBM or the vendor for your specific storage product to verify whether your controller or storage server supports the concurrent copy function.
Table space availability

If you specify COPY SHRLEVEL REFERENCE with the CONCURRENT option, and if you want to copy all of the data sets for a list of table spaces to the same dump data set, specify FILTERDDN in your COPY statement to improve table space availability. If you do not specify FILTERDDN, COPY might force DFSMSdss to process the list of table spaces sequentially, which might limit the availability of some of the table spaces that are being copied.

Related concepts:
- “How utilities restart with lists” on page 49
- Encrypted FlashCopy image copies, copies made with DFSMSdss concurrent copy, and system-level backups (Managing Security)

Specifying conditional image copies

Use the CHANGELIMIT option of the COPY utility to specify conditional image copies. You can use it to get a report of image copy information about a table space, or you can let Db2 decide whether to take an image copy based on this information.

You cannot use the CHANGELIMIT option for a table space or partition that is defined with TRACKMOD NO. If you change the TRACKMOD option from NO to YES, you must take an image copy before you can use the CHANGELIMIT option. When you change the TRACKMOD option from NO to YES for a nonpartitioned table space, you must take a full image copy by using DSNUM ALL before you can copy using the CHANGELIMIT option.

Obtaining image copy information about a table space

When you specify COPY CHANGELIMIT REPORTONLY, COPY reports image copy information for the table space and recommends the type of copy, if any, to take. The report includes:

- The total number of pages in the table space. This value is the number of pages that are to be copied if a full image copy is taken.
- The number of empty pages, if the table space is segmented.
- The number of changed pages. This value is the number of pages that are to be copied if an incremental image copy is taken.
- The percentage of changed pages.
- The type of image copy that is recommended.

Adding conditional code to your COPY job

You can add conditional code to your jobs so that an incremental or full image copy, or some other step, is performed depending on how much the table space has changed. For example, you can add a conditional MERGECOPY step to create a new full image copy if your COPY job took an incremental copy. COPY CHANGELIMIT uses the following return codes to indicate the degree that a table space or list of table spaces has changed:

1 (informational)
   If no CHANGELIMIT was met.

2 (informational)
   If the percentage of changed pages is greater than the low CHANGELIMIT and less than the high CHANGELIMIT value.
If the percentage of changed pages is greater than or equal to the high CHANGELIMIT value.

If you specify multiple COPY control statements in one job step, that job step reports the highest return code from all of the imbedded statements. Basically, the statement with the highest percentage of changed pages determines the return code and the recommended action for the entire list of COPY control statements that are contained in the subsequent job step.

**Using conditional copy with generation data groups (GDGs)**

For incremental image copies, COPY can avoid allocating empty image copy data sets if your real-time statistics are valid. However, even if your real-time statistics are not valid and you are using generation data groups (GDGs), you can ensure that COPY does not create any empty image copies by using the following procedure:

When you use generation data groups (GDGs) and need to make an incremental image copy, take the following steps to prevent creating an empty image copy:

1. Include in your job a first step in which you run COPY with CHANGELIMIT REPORTONLY. Set the SYSCOPY DD statement to DD DUMMY so that no output data set is allocated. If you specify REPORTONLY and use a template, Db2 does not dynamically allocate the data set.
2. Add a conditional JCL statement to examine the return code from the COPY CHANGELIMIT REPORTONLY step.
3. Add a second COPY step without CHANGELIMIT REPORTONLY to copy the table space or table space list based on the return code from the second step.

**Related information:**

"Allocation of sequential image copy data sets"

**Allocation of sequential image copy data sets**

If you specify the CHANGELIMIT option in the COPY utility control statement, COPY uses real-time statistics to determine when the criteria for CHANGELIMIT is met. If the CHANGELIMIT criteria is not met, COPY avoids allocating the image copy data set.

If you specify the FULL NO option in the COPY utility control statement, COPY uses real-time statistics to determine if a table space has changed pages since the last copy. If the table space has no changed pages, COPY avoids allocating the incremental copy data set.

If the real-time statistics are not valid, the object's space map pages are used to determine whether the CHANGELIMIT criteria is met or, for FULL NO, whether the table space has changed pages. Additionally, if real-time statistics are not used to determine whether the table space has changed pages, the image copy data set is always allocated.

**Related concepts:**

"Specifying conditional image copies“ on page 168

**Related reference:**

“Syntax and options of the COPY control statement” on page 135

**Related information:**
Preparing for recovery by using the COPY utility

To prepare for recovery, you can use the COPY utility to create copies and establish points of recovery.

Use the following guidelines to help you prepare for recovery:

- Consider periodically merging incremental image copies into one copy.
  The RECOVER utility merges all incremental image copies since the last full image copy, and it must have all the image copies available at the same time. If this requirement is likely to strain your system resources (for example, by demanding more tape units than are available), consider regularly merging multiple image copies into one copy.

Even if you do not periodically merge multiple image copies into one copy, when you do not have enough tape units, RECOVER can still attempt to recover the object. RECOVER dynamically allocates the full image copy and attempts to dynamically allocate all the incremental image copy data sets. If every incremental copy can be allocated, recovery proceeds to merge pages to table spaces and apply the log. If a point is reached where RECOVER cannot allocate an incremental copy, the log RBA of the last successfully allocated data set is noted. Attempts to allocate incremental copies cease, and the merge proceeds with only the allocated data sets. The log is applied from the noted RBA, and the incremental image copies that were not allocated are ignored.

- Create primary and backup image copies after a LOAD or REORG operation with LOG NO when an inline copy is not created. Create these copies, so that if the primary image copy is not available, fallback recovery with the secondary image copy is possible.

- If you use COPY SHRLEVEL REFERENCE to copy a list of referentially related structures, you do not need to quiesce these objects to create a consistent point of recovery. The copy serves as a point of consistency.

- For LOB data, quiesce and copy both the base table space and the LOB table space at the same time to establish a point of consistency for recovery. QUIESCE does not create a recovery point for a LOB table space that contains LOBs that are defined with LOG NO.

- If an index is in informational COPY-pending (ICOPY) status, take a full image copy of the index space so that the RECOVER utility can recover the index space.

  For an index that was defined with the COPY YES attribute the following utilities can place the index in ICOPY status:
  - REORG INDEX
  - REORG TABLESPACE LOG YES or NO
  - LOAD TABLE LOG YES or NO
  - REBUILD INDEX

  After the utility processing completes, take the full image copy. If you need to recover an index of which you did not take a full image copy, use the REBUILD INDEX utility to rebuild the index from data in the table space.

- Take image copies of table spaces with the NOT LOGGED attribute that have been updated since the last full copy.

  These table spaces are placed in ICOPY status. To copy the table spaces that have been updated, run the COPY utility with the SCOPE PENDING option.

Related concepts:

“Point-in-time recovery” on page 495
Improving performance of the COPY utility

You can improve the performance of the COPY utility by taking certain actions.

You can copy a list of objects in parallel to improve performance. Specifying a list of objects along with the SHRLEVEL REFERENCE option creates a single recovery point for that list of objects. Specifying the PARALLEL keyword allows you to copy a list of objects in parallel, rather than serially.

To calculate the number of threads you need when you specify the PARALLEL keyword, use the formula \( (n \times 2 + 1) \), where \( n \) is the number of objects that are to be processed in parallel, regardless of the total number of objects in the list. If you do not use the PARALLEL keyword, \( n \) is one and COPY uses three threads for a single-object COPY job.

You can merge a full image copy and subsequent incremental image copies into a new full copy by running the MERGECOPY utility. After reorganizing a table space, the first image copy must be a full image copy.

Do not base the decision of whether to run a full image copy or an incremental image copy on the number of rows that are updated since the last image copy was taken. Instead, base your decision on the percentage of pages that contain at least one updated record (not the number of updated records). Regardless of the size of the table, if more than 50% of the pages contain updated records, use full image copy (this saves the cost of a subsequent MERGECOPY). To find the percentage of changed pages, you can execute COPY with the CHANGELIMIT REPORTONLY option. Alternatively, you can execute COPY CHANGELIMIT to allow COPY to determine whether a full image copy or incremental copy is required.

Using Db2 data compression for table spaces can improve COPY performance because COPY does not decompress data. The performance improvement is proportional to the amount of compression.

Generation data group definitions for the COPY utility

Use generation data groups to hold image copies. Use of generation data groups offers the benefit of automating the allocation of data set names and the deletion of the oldest data set.

When you define the generation data group:

- You can specify that the oldest data set is automatically deleted when the maximum number of data sets is reached. If you do that, make the maximum number large enough to support all recovery requirements. When data sets are deleted, use the MODIFY utility to delete the corresponding rows in SYSIBM.SYSCOPY.
- Make the limit number of generation data sets equal to the number of copies that you want to keep. Use NOEMPTY to avoid deleting all the data sets from the integrated catalog facility catalog when the limit is reached.

Recommendation: Use templates when using generation data groups.

Related concepts:

"Specifying conditional image copies" on page 168
Using Db2 with DFSMS products

You can use Db2 with DFSMS products.

If image copy data sets are managed by HSM or SMS, all data sets are cataloged.

If you plan to use SMS, catalog all image copies. Never maintain cataloged and uncataloged image copies that have the same name.

Image copies on tape

Do not combine a full image copy and incremental image copies for the same table space on one tape volume. If you do, the RECOVER TABLESPACE utility cannot allocate the incremental image copies.

Termination of COPY

You can terminate an active or stopped COPY job with the TERM UTILITY command.

SYSIBM.SYSCOPY records

When you terminate COPY, Db2 inserts an ICTYPE=T record in the SYSIBM.SYSCOPY catalog table for each object that COPY started processing, but did not yet complete processing. The exception is if COPY is already in the UTILTERM phase. In this case, the image copy is considered complete.

For copies that are made with the SHRLEVEL REFERENCE option specified, some objects in the list might not have an ICTYPE=T record. For copies that were made with the SHRLEVEL CHANGE option specified, some objects might have a valid ICTYPE=F, I, or T record, or no record at all.

COPY does not allow you to take an incremental image copy if an ICTYPE=T record exists. You must take a full image copy.

Image copy data sets with DISP=(MOD,CATLG,CATLG)

If you terminate a COPY job where the data sets for the image copies are defined with parameter DISP=(MOD,CATLG,CATLG), take one of the following actions:

- If the job step contains only one COPY utility statement, delete the image copy data set. In this case, no row was written to SYSIBM.SYSCOPY, but an image copy data set was created and cataloged in the ICF catalog. You should delete that data set.

- If the job step contains several COPY utility statements, delete all image copy data sets that are not recorded in SYSIBM.SYSCOPY. In this case, a row for each successfully completed copy was written to SYSIBM.SYSCOPY. However, all the image copy data sets were created and cataloged. You should delete those data sets that are not recorded in SYSIBM.SYSCOPY.
Restart of COPY

You can restart the COPY utility.

**Recommendation:** Use restart current when possible, because it:
- Is valid for full image copies and incremental copies
- Is valid for a single job step with several COPY control statements
- Is valid for a list of objects
- Requires a minimum of re-processing
- Keeps the Db2 catalog and the integrated catalog facility catalog synchronized

If you do **not** use the **TERM UTILITY** command, you can restart a COPY job. COPY jobs with the CONCURRENT option restart from the beginning, and other COPY jobs restart from the last commit point. You cannot use RESTART(PHASE) for any COPY job. If you are restarting a COPY job with uncataloged output data sets, you must specify the appropriate volumes for the job in the JCL or on the TEMPLATE utility statement. Doing so could impact your ability to use implicit restart.

If the recommended procedure is not followed an ABEND 413-1C may occur during restart of the COPY.

**Restarting with a new data set**

If you define a new output data set for a current restart, complete the following actions before restarting the COPY job:
1. Copy the failed COPY output to the new data set.
2. Delete the old data set.
3. Rename the new data set to use the old data set name.

**Restarting COPY after an out-of-space condition**

You can also restart COPY from the last commit point after receiving an out-of-space condition.

**Related tasks:**
- “Restarting an online utility” on page 44
- “Restarting a utility after the output data set is full” on page 48

**Sample COPY control statements**

Use the sample control statements as models for developing your own COPY control statements.

In some cases, you might run a COPY utility job more than once. To avoid duplicate image copy data sets, a DSN qualifier is used in the following examples.

**Example 1: Making a full image copy**

The following control statement specifies that the COPY utility is to make a full image copy of table space DSN8D11A.DSN8S11E. The copy is to be written to the data set that is defined by the SYSCOPY DD statement in the JCL; SYSCOPY is the default.
Instead of defining the data sets in the JCL, you can use templates. In the following example, the preceding job is modified to use a template. In this example, the name of the template is LOCALDDN. The LOCALDDN template is identified in the COPY statement by the COPYDDN option.

```bash
//STEP1 EXEC DSNUPROC,UID='IUJMU111.COPYTS',
// UTPROC='',
// SYSTEM='DSN'
//SYSDDN DD DSN=COPIY001F.IFDY01,UNIT=SYSDA,VOL=SER=CPY01I,
// SPACE=((15,1)),DISP=(NEW,CATLG,CATLG)
//SYSIN DD *
COPY TABLESPACE DSN8D11A.DSN8S11E
/*
TEMPLATE LOCALDDN UNIT SYSDA DSN(COPY001F.IFDY01)
SPACE((15,1)) CYL DISP(NEW,CATLG,CATLG)
COPY TABLESPACE DSN8D11A.DSN8S11E COPYDDN(LOCALDDN)
/*
```

**Recommendation:** When possible, use templates to allocate data sets.

### Example 2: Making full image copies for local site and recovery site

The following COPY control statement specifies that COPY is to make primary and backup full image copies of table space DSN8D11P.DSN8S11C at both the local site and the recovery site. The COPYDDN option specifies the output data sets for the local site, and the RECOVERYDDN option specifies the output data sets for the recovery site. The PARALLEL option indicates that up to 2 objects are to be processed in parallel.

```bash
OPTIONS EVENT(ITEMERROR,SKIP)
COPY TABLESPACE DSN8D01P.DSN8S01C
  COPYDDN(COPY1,COPY2)
  RECOVERYDDN(COPY3,COPY4)
  PARALLEL(2)
```

### Example 3: Making full image copies of a list of objects

The control statement below specifies that COPY is to make local and recovery full image copies (both primary and backup) of the following objects:

- Table space DSN8D11A.DSN8S11D, and its indexes:
  - DSN8B10.XDEPT1
  - DSN8B10.XDEPT2
  - DSN8B10.XDEPT3
- Table space DSN8D11A.DSN8S11E, and its indexes:
  - DSN8710.XEMP1
  - DSN8710.XEMP2
These copies are to be written to the data sets that are identified by the COPYDDN and RECOVERYDDN options for each object. The COPYDDN option specifies the data sets for the copies at the local site, and the RECOVERYDDN option specifies the data sets for the copies at the recovery site. The first parameter of each of these options specifies the data set for the primary copy, and the second parameter specifies the data set for the backup copy. For example, the primary copy of table space DSN8D81A.DSN8S81D at the recovery site is to be written to the data set that is identified by the COPY3 DD statement.

PARALLEL(4) indicates that up to four of these objects can be processed in parallel. As the COPY job of an object completes, the next object in the list begins processing in parallel until all of the objects have been processed.

SHRLEVEL REFERENCE specifies that no updates are allowed during the COPY job. This option is the default and is recommended to ensure the integrity of the data in the image copy.

```plaintext
//STEP1 EXEC DSNUPROC,UID='IUJMU111.COPYTS',
  // UTPROC='',
  // SYSTEM='DSN'
//COPY1 DD DSN=C81A.S20001.D2003142.T155241.LP,
// SPACE=(CYL,(15,1)),DISP=(NEW,CATLG,CATLG)
//COPY2 DD DSN=C81A.S20001.D2003142.T155241.LB,
// SPACE=(CYL,(15,1)),DISP=(NEW,CATLG,CATLG)
//COPY3 DD DSN=C81A.S20001.D2003142.T155241.RP,
// SPACE=(CYL,(15,1)),DISP=(NEW,CATLG,CATLG)
//COPY4 DD DSN=C81A.S20001.D2003142.T155241.RB,
// SPACE=(CYL,(15,1)),DISP=(NEW,CATLG,CATLG)
//COPY5 DD DSN=C81A.S20002.D2003142.T155241.LP,
// SPACE=(CYL,(15,1)),DISP=(NEW,CATLG,CATLG)
//COPY6 DD DSN=C81A.S20002.D2003142.T155241.LB,
// SPACE=(CYL,(15,1)),DISP=(NEW,CATLG,CATLG)
//COPY7 DD DSN=C81A.S20002.D2003142.T155241.RP,
// SPACE=(CYL,(15,1)),DISP=(NEW,CATLG,CATLG)
//COPY8 DD DSN=C81A.S20002.D2003142.T155241.RB,
// SPACE=(CYL,(15,1)),DISP=(NEW,CATLG,CATLG)
//COPY9 DD DSN=C81A.S20003.D2003142.T155241.LP,
// SPACE=(CYL,(15,1)),DISP=(NEW,CATLG,CATLG)
//COPY10 DD DSN=C81A.S20003.D2003142.T155241.LB,
// SPACE=(CYL,(15,1)),DISP=(NEW,CATLG,CATLG)
//COPY11 DD DSN=C81A.S20003.D2003142.T155241.RP,
// SPACE=(CYL,(15,1)),DISP=(NEW,CATLG,CATLG)
//COPY12 DD DSN=C81A.S00003.D2003142.T155241.RB,
// SPACE=(CYL,(15,1)),DISP=(NEW,CATLG,CATLG)
//COPY13 DD DSN=C81A.S00004.D2003142.T155241.LP,
// SPACE=(CYL,(15,1)),DISP=(NEW,CATLG,CATLG)
//COPY14 DD DSN=C81A.S00004.D2003142.T155241.LB,
```
You can also write this COPY job so that it uses lists and templates, as shown below. In this example, the name of the template is T1. Note that this TEMPLATE statement does not contain any space specifications for the dynamically allocated data sets. Instead, Db2 determines the space requirements. The T1 template is identified in the COPY statement by the COPYDDN and RECOVERYDDN options. The name of the list is COPYLIST. This list is identified in the COPY control statement by the LIST option.
Example 4: Using template switching

The following TEMPLATE control statement assumes that tables space SMALL.TS occupies 10 cylinders and table space LARGE.TS occupies 100 cylinders. Both COPY statements use the SMALLTP template which specifies a limit of 20 cylinders. Table space SMALL.TS is smaller than this limit so no switching is performed. The output data set for table space SMALL.TS will be allocated on UNIT=SYSALLDA. Table space LARGE.TS is larger than this limit so the template is switched to the LARGETP template. The output data set for table space LARGE.TS will be allocated on UNIT=TAPE.

Note that the DSN option of the TEMPLATE statement identifies the names of the data sets to which the copies are to be written.

Each of the preceding COPY jobs create a point of consistency for the table spaces and their indexes. You can subsequently use the RECOVER utility with the TOLOGPOINT option to recover all of these objects.

Example 5: Making full image copies of a list of objects in parallel on tape

The following COPY control statement specifies that COPY is to make image copies of the specified table spaces and their associated index spaces in parallel and stack the copies on different tape devices.

The PARALLEL 2 option specifies that up to two objects can be processed in parallel. The TAPEUNITS 2 option specifies that up to two tape devices can be dynamically allocated at one time. The COPYDDN option for each object specifies the data set that is to be used for the local image copy. In this example, all of these data sets are dynamically allocated and defined by templates. For example, table space DSN8D81A.DSN8S81D is copied into a data set that is defined by the A1 template.
The TEMPLA TE utility control statements define the templates A1 and A2.

```
//COPY2A EXEC DSNUPROC,SYSTEM=DSN
//SYSIN   DD *
   TEMPLATE A1 DSN(&DB..&SP..COPY1) UNIT CART STACK YES
   TEMPLATE A2 DSN(&DB..&SP..COPY2) UNIT CART STACK YES
COPY PARALLEL 2 TAPEUNITS 2
   TABLESPACE DSN8D81A.DSN8S81D COPYDDN(A1)
   INDEXSPACE DSN8B10.XDEPT COPYDDN(A1)
   TABLESPACE DSN8D81A.DSN8S81E COPYDDN(A2)
   INDEXSPACE DSN8B10.YDEPT COPYDDN(A2)
```

Although use of templates is recommended, you can also define the output data sets by coding JCL DD statements, as in the following example. This COPY control statement also specifies a list of objects to be processed in parallel, but in this case, the data sets are defined by DD statements. In each DD statement, notice the parameters for the VOLUME option. These values show that the data sets are defined on three different tape devices as follows:

- The first tape device contains data sets that are defined by DD statements DD1 and DD4. (For DD4, the VOLUME option has a value of *.DD1 for the REF parameter.)
- A second tape device contains data sets that are defined by DD statements DD2 and DD3. (For DD3, the VOLUME option has a value of *.DD3 for the REF parameter.)
- A third tape device contains the data set that is defined by DD statement DD5.

The following table spaces are to be processed in parallel on two different tape devices:

- DSN8D81A.DSN8S81D on the device that is defined by the DD1 DD statement and the device that is defined by the DD5 DD statement
- DSN8D81A.DSN8S81E on the device that is defined by the DD2 DD statement

Copying of the following tables spaces must wait until processing has completed for DSN8D81A.DSN8S81D and DSN8D81A.DSN8S81E:

- DSN8D81A.DSN8S81F on the device that is defined by the DD2 DD statement after DSN8D81A.DSN8S81E completes processing
- DSN8D81A.DSN8S81G on the device that is defined by the DD1 DD statement after DSN8D81A.DSN8S81D completes processing
Example 6: Using both JCL-defined and template-defined data sets to copy a list of objects on tape

This example uses both JCL DD statements and utility templates to define four data sets for the image copies. The JCL defines two data sets (DB1.TS1.CLP and DB2.TS2.CLB.BACKUP), and the TEMPLATE utility control statements define two data sets that are to be dynamically allocated (&DB..&SP.COPY1 and &DB..&SP.COPY2).

The COPYDDN options in the COPY control statement specify the data sets that are to be used for the local primary and backup image copies of the specified table spaces. For example, the primary copy of table space DSN8D81A.DSN8S71D is to be written to the data set that is defined by the DD1 DD statement (DB1.TS1.CLP), and the primary copy of table space DSN8D81A.DSN8S71E is to be written to the data set that is defined by the A1 template (&DB..&SP.COPY1).

Four tape devices are allocated for this COPY job: the JCL allocates two tape drives, and the TAPEUNITS 2 option in the COPY statement indicates that two tape devices are to be dynamically allocated. Note that the TAPEUNITS option applies only to those tape devices that are dynamically allocated by the TEMPLATE statement.

Recommendation: Although this example shows how to use both templates and DD statements, use only templates, if possible.
Example 7: Using LISTDEF to define a list of objects to copy in parallel to tape

The following example uses the LISTDEF utility to define a list of objects to be copied in parallel to different tape sources. The COPY control statement specifies that the table spaces that are included in the PAYROLL list are to be copied. (The PAYROLL list is defined by the LISTDEF control statement.) The TEMPLATE control statements define two output data sets, one for the local primary copy (&DB..&COPY..LOCAL) and one for the recovery primary copy (&DB..&COPY..REMOTE).

```
//COPY1D EXEC DSNUPROC,SYSTEM=DSN
//DD1 DD DSN=DB1.TS1.CLP,
  // DISP=(,CATLG),
  // UNIT=3490, LABEL=(1,SL)
  // VOLUME=(,RETAIN)
//DD2 DD DSN=DB2.TS2.CLB.BACKUP,
  // DISP=(,CATLG),
  // UNIT=3490, LABEL=(2,SL)
  // VOLUME=(,RETAIN)
//SYSIN DD *
  TEMPLATE A1 DSN(&DB..&SN..COPY1) UNIT CART STACK YES
  TEMPLATE A2 DSN(&DB..&SN..COPY2) UNIT CART STACK YES
COPY PARALLEL 2 TAPEUNITS 2
  TABLESPACE DSN8D81A.DSN8S81D COPYDDN(DD1,DD2)
  TABLESPACE DSN8D81A.DSN8S81E COPYDDN(A1,A2)
```

Figure 16. Example of using both JCL-defined and template-defined data sets to copy a list of objects on a tape

In the preceding example, the utility determines the number of tape streams to use by dividing the value for TAPEUNITS (8) by the number of output data sets (2) for a total of 4 in this example. For each tape stream, the utility attaches one subtask. The list of objects is sorted by size and processed in descending order. The first subtask to finish processes the next object in the list. In this example, the PARALLEL(10) option limits the number of objects to be processed in parallel to 10 and attaches four subtasks. Each subtask copies the objects in the list in parallel to two tape drives, one for the primary and one for the recovery output data sets.

Example 8: Making incremental copies with updates allowed

The FULL NO option in the following COPY control statement specifies that COPY is to make incremental image copies of any specified objects. In this case, the objects to be copied are those objects that are included in the NAME1 list, as indicated by the LIST option. The preceding LISTDEF utility control statement defines the NAME1 list to include index space DSN8D81A.XEMP1 and table space DSN8D81A.DSN8S81D. Although one of the objects to be copied is an index space and COPY does not take incremental image copies of index spaces, the job does not fail; COPY takes a full image copy of the index space instead. However, if a COPY FULL NO statement identifies only an index that is not part of a list, the COPY job fails.
All specified copies (local primary and backup copies and remote primary and backup copies) are written to data sets that are dynamically allocated according to the specifications of the COPYDS template. This template is defined in the preceding TEMPLATE utility control statement.

The SHRLEVEL CHANGE option in the following COPY control statement specifies that updates can be made during the COPY job.

```
TEMPLATE COPYDS DSN &US.2.&SN..&LR..&PB..&DATE.
   LISTDEF NAME1 INCLUDE INDEXSPACE DSN8D81A.XEMP1
       INCLUDE TABLESPACE DSN8D81A.DSN8S81D
   COPY LIST NAME1 COPYDDN(COPYDS, COPYDS) RECOVERYDDN(COPYDS, COPYDS)
   FULL NO SHRLEVEL CHANGE
```

**Example 9: Making a conditional image copy**

The CHANGELIMIT(5) option in the following control statement specifies the following conditions for making an image copy of table space DSN8D11P.DSN8S11C:

- Take a full image copy of the table space if the percentage of changed pages is equal to or greater than 5%.
- Take an incremental image copy of the table space if the percentage of changed pages is greater than 0 and less than 5%.
- Do not take an image copy if no pages have changed.

```
COPY TABLESPACE DSN8D11P.DSN8S11C CHANGELIMIT(5)
```

**Example 10: Reporting image copy information for a table space**

The REPORTONLY option in the following control statement specifies that image copy information is to be displayed only; no image copies are to be made. The CHANGELIMIT(10,40) option specifies that the following information is to be displayed:

- Recommendation that a full image copy be made if the percentage of changed pages is equal to or greater than 40%.
- Recommendation that an incremental image copy be made if the percentage of changed pages is greater than 10% and less than 40%.
- Recommendation that no image copy be made if the percentage of changed pages is 10% or less.

```
COPY TABLESPACE DSN8D11P.DSN8S11C CHANGELIMIT(10,40) REPORTONLY
```

**Example 11: Invoking DFSMSdss concurrent copy**

The CONCURRENT option in the following COPY control statement specifies that DFSMSdss concurrent copy is to make a full image copy of the objects in the COPYLIST list (table space DSN8D81A.DSN8S81D and table space DSN8D81A.DSN8S81P). The COPYDDN option indicates that the copy is to be written to the data set that is defined by the SYSCOPY1 template. The DSSPRINT DD statement specifies the data set for message output.
Example 12: Invoking DFSMSdss concurrent copy and using a filter data set

The control statement specifies that DFSMSdss concurrent copy is to make full image copies of the objects in the TSLIST list (table spaces TS1, TS2, and TS3). The FILTERDDN option specifies that COPY is to use the filter data set that is defined by the FILT template. All output is sent to the SYSCOPY data set, as indicated by the COPYDDN(SYSCOPY) option. SYSCOPY is the default. This data set is defined in the preceding TEMPLATE control statement.

```
//COPY EXEC DSNUPROC,SYSTEM=DSN
//SYSPRINT DD DSN=COPY1.PRINT1,DISP=(NEW,CATLG,CATLG),
//   SPACE=(4000,(20,20),,,ROUND),UNIT=SYSDA,Vol=SER=DB2CC5
//DSSPRINT DD DSN=COPY1.PRINT2,DISP=(NEW,CATLG,CATLG),
//   SPACE=(4000,(20,20),,,ROUND),UNIT=SYSDA,Vol=SER=DB2CC5
//SYSIN DD *
   TEMPLATE SYSCOPY1 DSN &DB..&TS..COPY&IC.&LR.&PB..D&DATE..T&TIME.
   UNIT(SYSDA) DISP (MOD,CATLG,CATLG)
   LISTDEF COPYLIST INCLUDE TABLESPACE DSN8D81A.DSN8S81D
   INCLUDE TABLESPACE DSN8D81A.DSN8S81P
COPY LIST COPYLIST
COPYDDN (SYSCOPY1)
CONCURRENT
```

Figure 17. Example of invoking DFSMSdss concurrent copy with the COPY utility

Example 13: Copying LOB table spaces together with related objects

Assume that table space TPIQU01 is a base table space and that table spaces TLIQU0A1, TLIQU0A2, TLIQU0A3, and TLIQU0A4 are LOB table spaces. The control statement specifies that COPY is to take the following actions:

- Take a full image copy of each specified table space if the percentage of changed pages is equal to or greater than the highest decimal percentage value for the CHANGELIMIT option for that table space. For example, if the percentage of changed pages for table space TPIQU01 is equal to or greater than 6.7%, COPY is to take a full image copy.

- Take an incremental image copy of each specified table space if the percentage of changed pages is in the range between the specified decimal percentage values for the CHANGELIMIT option for that table space. For example, if the percentage of changed pages for table space TLIQU0A1 is greater than 7.9% and less than 25.3%, COPY is to take an incremental image copy.

- Do not take an image copy of each specified table space if the percentage of changed pages is equal to or less than the lowest decimal percentage value for
the CHANGELIMIT option for that table space. For example, if the percentage of changed pages for table space TLIQUDA2 is equal to or less than 2.2%, COPY is not to take an incremental image copy.

- Take full image copies of index spaces IPIQUD01, IXIQUD02, IUIQUD03, IXIQUDA1, IXIQUDA2, IXIQUDA3, and IXIQUDA4.

```
COPY
TABLESPACE DBIQUD01.TPIQUD01 DSNUM ALL CHANGELIMIT(3.3,6.7)
COPYDDN(COPYTB1)
TABLESPACE DBIQUD01.TLIQUDA1 DSNUM ALL CHANGELIMIT(7.9,25.3)
COPYDDN(COPYTA1)
TABLESPACE DBIQUD01.TLIQUDA2 DSNUM ALL CHANGELIMIT(2.2,4.3)
TABLESPACE DBIQUD01.TLIQUDA3 DSNUM ALL CHANGELIMIT(1.2,9.3)
TABLESPACE DBIQUD01.TLIQUDA4 DSNUM ALL CHANGELIMIT(2.2,4.0)
TABLESPACE DBIQUD01.TLIQUDA1 DSNUM ALL CHANGELIMIT(7.9,25.3)
TABLESPACE DBIQUD01.TLIQUDA2 DSNUM ALL CHANGELIMIT(2.2,4.3)
TABLESPACE DBIQUD01.TLIQUDA3 DSNUM ALL CHANGELIMIT(1.2,9.3)
TABLESPACE DBIQUD01.TLIQUDA4 DSNUM ALL CHANGELIMIT(2.2,4.0)
INDEXSPACE DBIQUD01.IPIQUD01 DSNUM ALL
COPYDDN(COPYIX1)
INDEXSPACE DBIQUD01.IXIQUDA2 DSNUM ALL
COPYDDN(COPYIX2)
INDEXSPACE DBIQUD01.IXIQUDA3 DSNUM ALL
COPYDDN(COPYIX3)
INDEXSPACE DBIQUD01.IXIQUDA1 DSNUM ALL
COPYDDN(COPYIX1A)
INDEXSPACE DBIQUD01.IXIQUDA2 DSNUM ALL
COPYDDN(COPYIX2A)
INDEXSPACE DBIQUD01.IXIQUDA3 DSNUM ALL
COPYDDN(COPYIX3A)
INDEXSPACE DBIQUD01.IXIQUDA4 DSNUM ALL
COPYDDN(COPYIX4A)
SHRLEVEL REFERENCE
```

Figure 19. Example of copying LOB table spaces together with related objects

**Example 14: Using GDGs to make a full image copy**

The following control statement specifies that the COPY utility is to make a full image copy of table space DBLT2501.TPLT2501. The local copies are to be written to data sets that are dynamically allocated according to the COPYTEM1 template. The remote copies are to be written to data sets that are dynamically allocated according to the COPYTEM2 template. For both of these templates, the DSN option indicates the name of generation data group JULTU225 and the generation number of +1. (If a GDG base does not already exist, Db2 creates one.) Both of these output data sets are to be modeled after the JULTU225.MODEL data set (as indicated by the MODELDCB option in the TEMPLATE statements).

```
//***********************************************************
//* COMMENT: MAKE A FULL IMAGE COPY OF THE TABLESPACE.
//* USE A TEMPLATE FOR THE GDG.
//***********************************************************
//STEP2 EXEC DSNUPROC,UID='JULTU225.COPY',
//  UTPROC='',
//  SYSTEM='SSTR'
//SYSIN DD *
TEMPLATE COPYTEM1
  UNIT SYSDA
  DSN 'JULTU225.GDG.LOCAL.&PB.(+1)'
  MODELDCB JULTU225.MODEL
TEMPLATE COPYTEM2
  UNIT SYSDA
  DSN 'JULTU225.GDG.REMOTE.&PB.(+1)'
  MODELDCB JULTU225.MODEL
COPY TABLESPACE DBLT2501.TPLT2501
```
Example 15: Copying updated table space data

The following control statement indicates that COPY is to copy only the objects that have been updated. SCOPE PENDING indicates that you want to copy only those objects in COPY-pending or informational COPY-pending status.

COPY SHRLEVEL REFERENCE
  TABLESPACE DBIQUD01.TPIQUD01 DSNUM ALL CHANGETLIMIT(3.3,6.7)
  COPYDDN(COPYTB1)
  TABLESPACE DBIQUD01.TLIQUA1 DSNUM ALL CHANGETLIMIT(7.9,25.3)
  COPYDDN(COPYTA1)
  TABLESPACE DBIQUD01.TLIQUA2 DSNUM ALL CHANGETLIMIT(2.2,4.3)
  COPYDDN(COPYTA2)
  TABLESPACE DBIQUD01.TLIQUA3 DSNUM ALL CHANGETLIMIT(1.2,9.3)
  COPYDDN(COPYTA3)
  TABLESPACE DBIQUD01.TLIQUA4 DSNUM ALL CHANGETLIMIT(2.2,4.0)
  COPYDDN(COPYTA4)
  INDEXSPACE DBIQUD01.IPIQUD01 DSNUM ALL
  COPYDDN(COPYIX1) PARALLEL(4)
  SCOPE PENDING
/*

Example 16: Copying clone table data

The following control statement indicates that COPY is to copy only clone table data in the specified table spaces or indexes.

COPY SHRLEVEL REFERENCE CLONE
  TABLESPACE DBIQUD01.TPIQUD01 DSNUM ALL CHANGETLIMIT(3.3,6.7)
  COPYDDN(COPYTB1)
  TABLESPACE DBIQUD01.TLIQUA1 DSNUM ALL CHANGETLIMIT(7.9,25.3)
  COPYDDN(COPYTA1)
  TABLESPACE DBIQUD01.TLIQUA2 DSNUM ALL CHANGETLIMIT(2.2,4.3)
  COPYDDN(COPYTA2)
  TABLESPACE DBIQUD01.TLIQUA3 DSNUM ALL CHANGETLIMIT(1.2,9.3)
  COPYDDN(COPYTA3)
  TABLESPACE DBIQUD01.TLIQUA4 DSNUM ALL CHANGETLIMIT(2.2,4.0)
  COPYDDN(COPYTA4)
  INDEXSPACE DBIQUD01.IPIQUD01 DSNUM ALL
  COPYDDN(COPYIX1)
Chapter 15. COPYTOCOPY

The COPYTOCOPY online utility makes image copies from an image copy that was taken by the COPY utility. The COPYTOCOPY utility can also make image copies from inline copies that the REORG or LOAD utilities make.

Starting with the local primary copy or a recovery-site primary copy, or a copy created by using FlashCopy technology, COPYTOCOPY can make up to four sequential format copies of one or more of the following types of copies:

- Local primary
- Local backup
- Recovery site primary
- Recovery site backup

You can make both full and incremental image copies of a LOB or XML table space.

You cannot run COPYTOCOPY on concurrent copies.

The RECOVER utility uses the copies when recovering a table space or index space to the most recent time or to a previous time. These copies can also be used by MERGECOPY, UNLOAD, and possibly a subsequent COPYTOCOPY execution.

Output

Output from the COPYTOCOPY utility consists of:

- Up to three sequential data sets that contain the image copy. If the copy base is a FlashCopy, four sequential copies can be made.
- Up to three sequential data sets that contain the image copy.
- Rows in the SYSIBM.SYSCOPY catalog table that describe the image copy data sets that are available to the RECOVER utility. Your installations responsible for ensuring that these data sets are available if the RECOVER utility requests them.

The entries for SYSCOPY columns remain the same as the original entries in the SYSCOPY row when the COPY utility recorded them. The COPYTOCOPY job inserts values in the columns DSNAME, GROUP_MEMBER, JOBNAME, AUTHID, DSVOLSER, and DEVTYPE.

Authorization required

To execute this utility, you must use a privilege set that includes one of the following authorities:

- IMAGCOPY privilege for the database
- DBADM, DBCTRL, or DBMAINT authority for the database. If the object on which the utility operates is in an implicitly created database, DBADM authority on the implicitly created database or DSNDB04 is required.
- System DBADM authority
- SYSCTRL or SYSADM authority

An ID with installation SYSOPR authority can also execute COPYTOCOPY, but only on a table space in the DSNDB01 or DSNDB06 database.
Restrictions on running COPYTOCOPY

- COPYTOCOPY does not support the following catalog and directory objects:
  - DSNDB01.SYSUTILX and its indexes
  - DSNDB01.DBD01 and its indexes
  - DSNDB01.SYSDBDXA and its indexes
  - DSNDB06.SYSTSCPY and its indexes
- An image copy from a COPY job with the CONCURRENT option cannot be processed by COPYTOCOPY.
- COPYTOCOPY does not check the recoverability of an object.
- COPYTOCOPY cannot be run on a table space during the period after RECOVER is run to a point in time before materialization of pending definition changes and before REORG is run to complete the point-in-time recovery process.
- COPYTOCOPY does not support a range of partitions within a partitioned table space. Specify individual DSNUM(n). From the inline copy, COPYTOCOPY copies only the specified partition into the output image copy data set.

Execution phases of COPYTOCOPY

The COPYTOCOPY utility operates in these phases:

<table>
<thead>
<tr>
<th>Phase</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTILINIT</td>
<td>Performs initialization</td>
</tr>
<tr>
<td>CPY2CPY</td>
<td>Copies an image copy</td>
</tr>
<tr>
<td>UTILTERM</td>
<td>Performs cleanup</td>
</tr>
</tbody>
</table>

Syntax and options of the COPYTOCOPY control statement

The COPYTOCOPY utility control statement, with its multiple options, defines the function that the utility job performs.

You can create a control statement with the ISPF/PDF edit function. After creating it, save it in a sequential or partitioned data set. When you create the JCL for running the job, use the SYSIN DD statement to specify the name of the data set that contains the utility control statement.

Syntax diagram

```
COPYTOCOPY LIST listdef-name from-copy-spec data-set-spec CLONE
         ts-num-spec from-copy-spec data-set-spec
         index-name-spec from-copy-spec data-set-spec
```

ts-num-spec:
index-name-spec:

Notes:
1 INDEXSPACE is the preferred specification.
2 Not valid for nonpartitioning indexes.

from-copy-spec:

Notes:
1 Not valid with the INDEXSPACE or INDEX keyword.
2 Not valid with the LIST keyword.

data-set-spec:
Notes:

1. Use this option if you want to make a local site primary copy from one of the recovery site copies.
2. You can specify up to three DD names for both the COPYDDN and RECOVERYDDN options combined.

Option descriptions

LIST listdef-name
   Specifies the name of a previously defined LISTDEF list name. The utility allows one LIST keyword for each COPYTOCOPY control statement. Do not specify LIST with either the INDEX or TABLESPACE keywords. Db2 invokes COPYTOCOPY once for the entire list.

   This utility does not support lists that specify more than 32,000 objects.
   Partitions of table spaces or index spaces that are included by the PARTLEVEL keyword count as separate objects.

   This utility only processes clone data if the CLONE keyword is specified. The use of CLONE YES on the LISTDEF statement is not sufficient.
   The partitions or partition ranges can be specified in a list.

TABLESPACE database-name.tablespace-name
   Specifies the table space (and, optionally, the database it belongs to) that is to be copied. database-name is the name of the database that the table space belongs to. The default value is DSNDB04.
   table-space-name is the name of the table space to be copied.

INDEXSPACE database-name.index-space-name
   Specifies the qualified name of the index space that is to be copied; the name is obtained from the SYSIBM.SYSINDEXES table. Define the index space with the COPY YES attribute.
   database-name optionally specifies the name of the database that the index space belongs to. The default value is DSNDB04.
   index-space-name specifies the name of the index space that is to be copied.

INDEX creator-id.index-name
   Specifies the index that is to be copied. Enclose the index name in quotation marks if the name contains a blank.
   creator-id optionally specifies the creator of the index. The default value is the user identifier for the utility.
   index-name specifies the name of the index that is to be copied.
DSNUM
Identifies a partition or data set, within the table space or the index space, that is to be copied. The keyword ALL specifies that the entire table space or index space is to be copied.

You cannot specify DSNUM for nonpartitioned indexes.

ALL
Specifies that the entire table space or index space is to be copied. You must use ALL for a nonpartitioned secondary index.

integer
Is the number of a partition or data set that is to be copied.

An integer value is not valid for nonpartitioned secondary indexes.

For a partitioned table space or index space, the integer is its physical partition number. The maximum is 4096.

For a nonpartitioned table space, find the integer at the end of the data set name as cataloged in the VSAM catalog. The data set name has the following format:

catname.DSNDbx.dbname.spacename.y000Z.Annn

In this format:

catname
Is the VSAM catalog name or alias.

x
Is C or D.

dbname
Is the database name.

spacename
Is the table space or index space name.

y
Is I or J.

z
Is 1 or 2.

nnn
Is the data set integer.

Specifying or using the default of DSNUM(ALL) causes COPYTOCOPY to look for an input image copy that was taken at the entire table space or index space level.

FROMLASTCOPY
Specifies the most recent image copy that was taken for the table space or index space that is to be the input to the COPYTOCOPY utility. This could be a full image copy or incremental copy that is retrieved from SYSIBM.SYSCOPY.

FROMLASTFULLCOPY
Specifies the most recent full image copy that was taken for the object, which is to be the input to the COPYTOCOPY job.

FROMLASTINCRCOPY
Specifies the most recent incremental image copy that was taken for the object that is to be the input to COPYTOCOPY job.

FROMLASTINCRCOPY is not valid with the INDEXSPACE or INDEX keyword. If FROMLASTINCRCOPY is specified for an INDEXSPACE or INDEX, COPYTOCOPY uses the last full copy that was taken, if one is available.
**FROMLASTFLASHCOPY**

Specifies that the most recent FlashCopy image copy that was taken for the object is to be the input to COPYTOCOPY job.

**FROMCOPY dsn**

Specifies a particular image copy data set (dsn) as the input to the COPYTOCOPY job. This option is not valid for LIST.

If the image copy data set is a generation data set, then supply a fully qualified data set name, including the absolute generation and version number.

If the image copy data set is not a generation data set and more than one image copy data set have the same data set name, use the FROMVOLUME option to identify the data set exactly.

**FROMVOLUME**

Identifies the image copy data set.

**CATALOG**

Identifies the data set as cataloged. Use this option only for an image copy that was created as a cataloged data set. (Its volume serial is not recorded in SYSIBM.SYSCOPY.)

COPYTOCOPY refers to the SYSIBM.SYSCOPY catalog table during execution. If you use FROMVOLUME CATALOG, the data set must be cataloged. If you remove the data set from the catalog after creating it, you must catalog the data set again to make it consistent with the record that appears in SYSIBM.SYSCOPY for this copy.

**vol-ser**

Identifies the data set by an alphanumeric volume serial identifier of its first volume. Use this option only for an image copy that was created as a noncataloged data set. Specify the first vol-ser in the SYSCOPY record to locate a data set that is stored on multiple tape volumes. If an individual volume serial number contains leading zeros, it must be enclosed in single quotation marks.

**FROMSEQNO n**

Identifies the image copy data set by its file sequence number. n is the file sequence number.

**COPYDDN (ddname1,ddname2)**

Specifies a DD name (ddname) or a TEMPLATE name for the primary (ddname1) and backup (ddname2) copied data sets for the image copy at the local site. If ddname2 is specified by itself, COPYTOCOPY expects the local site primary image copy to exist. If it does not exist, error message DSNU1401 is issued and the process for the object is terminated.

**Recommendation:** Catalog all of your image copy data sets.

You cannot have duplicate image copy data sets. If the DD statement identifies a noncataloged data set with the same name, volume serial, and file sequence number as one that is already recorded in SYSIBM.SYSCOPY, COPYTOCOPY issues a message and no copy is made. If the DD statement identifies a cataloged data set with only the same name, no copy is made. For cataloged image copy data sets, you must specify CATLG for the normal termination disposition in the DD statement; for example, DISP=(MOD,CATLG,CATLG). The DSVOLSER field of the SYSCOPY entry is blank.

When the image copy data set is going to a tape volume, specify VOL=SER parameter in the DD statement.
The COPYDDN keyword specifies either a DD name or a TEMPLATE name specification from a previous TEMPLATE control statement. If utility processing detects that the specified name is both a DD name in the current job step and a TEMPLATE name, the utility uses the DD name.

**RECOVERYDDN (ddname3,ddname4)**

Specifies a DD name (ddname) or a TEMPLATE name for the primary (ddname3) and backup (ddname4) copied data sets for the image copy at the recovery site. If ddname4 is specified by itself, COPYTOCOPY expects the recovery site primary image copy to exist. If this image copy does not exist, error message DSNU1401 is issued and the process for the object is terminated.

You cannot have duplicate image copy data sets. The same rules apply for RECOVERYDDN as for COPYDDN.

The RECOVERYDDN keyword specifies either a DD name or a TEMPLATE name specification from a previous TEMPLATE control statement. If utility processing detects that the specified name is both a DD name in the current job step and a TEMPLATE name, the utility uses the DD name.

**CLONE**

Indicates that COPYTOCOPY is to process only image copy data sets that were taken against clone tables or indexes on clone tables. This utility will only process clone data if the CLONE keyword is specified. The use of CLONED YES on the LISTDEF statement is not sufficient.

Related reference:
- Chapter 34, “TEMPLATE,” on page 793
- Chapter 18, “LISTDEF,” on page 215

### Data sets that COPYTOCOPY uses

The COPYTOCOPY utility uses a number of data sets during its operation.

The following table describes the data sets that COPYTOCOPY uses. The table lists the DD name that is used to identify the data set, a description of the data set, and an indication of whether it is required. Include statements in your JCL for each required data set and any optional data sets that you want to use.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSIN</td>
<td>Input data set that contains the utility control statement.</td>
<td>Yes</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>Output data set for messages.</td>
<td>Yes</td>
</tr>
<tr>
<td>Output copies</td>
<td>From one to four output data sets that contain the resulting image copy data sets. Specify their DD names with the COPYDDN and RECOVERYDDN options of the utility control statement.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

The following objects are named in the utility control statement and do not require DD statements in the JCL:

**Table space or Index space**

Object that is to be copied. (If you want to copy only certain partitions in a partitioned table space, use the DSNUM option in the control statement.)
Db2 catalog objects

Objects in the catalog that COPYTOCOPY accesses. The utility records each copy in the Db2 catalog table SYIBM.SYSCOPY.

Input image copy data set

This information is accessed through the Db2 catalog. COPYTOCOPY retains all tape mounts for you. You do not need to code JCL statements to retain tape mounts. If the image copy data sets that are used by COPYTOCOPY reside on the same tape, you do not need to remove the tape.

Output data set size

Image copies are written to sequential non-VSAM data sets.

Recommendation: Use a template for the image copy data set for a table space by specifying a TEMPLATE statement without the SPACE keyword. When you omit this keyword, the utility calculates the appropriate size of the data set for you.

Alternatively, you can find the approximate size, in bytes, of the image copy data set for a table space by using the following procedure:

1. Find the high-allocated page number from the COPYPAGE SF column of SYIBM.SYSCOPY or from information in the VSAM catalog data set.
2. Multiply the high-allocated page number by the page size.

Another option is to look at the size of the input image copy.

JCL parameters: You can specify a block size for the output by using the BLKSIZE parameter on the DD statement for the output data set. Valid block sizes are multiples of 4096 bytes. It is recommended that the BLKSIZE parameter be omitted. The TAPEBLKSIZE parameter of the DEV$UP$xx member of SYS$1.PARMLIB controls the block size limit for tapes. See the z/OS MVS Initialization and Tuning Guide for more details.

Cataloging image copies

To catalog your image copy data sets, use the DISP=(NEW,CATLG,CATLG) parameter in the DD statement or TEMPLATE that is named by the COPYDDN or RECOVERYDDN option. After the image copy is taken, the DSVOLSER column of the row that is inserted into SYIBM.SYSCOPY contains blanks.

Duplicate image copy data sets are not allowed. If a cataloged data set is already recorded in SYIBM.SYSCOPY with the same name as the new image copy data set, a message is issued and the copy is not made.

When RECOVER locates the entry in SYIBM.SYSCOPY, it uses the ICF catalog to allocate the required data set. If you have uncataloged the data set, the allocation fails. In that case, the recovery can still go forward; RECOVER searches for a previous image copy. But even if RECOVER finds one, it must use correspondingly more of the log to recover. You are responsible for keeping the z/OS catalog consistent with SYIBM.SYSCOPY with regard to existing image copy data sets.

Related concepts:

“Data sets that online utilities use” on page 16
Concurrency and compatibility for COPYTOCOPY

The COPYTOCOPY utility has certain concurrency and compatibility characteristics associated with it.

Db2 treats individual data and index partitions as distinct target objects. Utilities that operate on different partitions of the same table space or index space are compatible.

Claims

The following table shows which claim classes COPYTOCOPY claims on the target object.

Table 22. Claim classes of COPYTOCOPY operations.

<table>
<thead>
<tr>
<th>Target</th>
<th>COPYTOCOPY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table space or partition, or index space or partition</td>
<td>UTRW</td>
</tr>
</tbody>
</table>

Legend:
- UTRW - Utility restrictive state - read-write access allowed

Compatibility

The following table documents which utilities can run concurrently with COPYTOCOPY on the same target object. The target object can be a table space, an index space, or a partition of a table space or index space. If compatibility depends on particular options of a utility, that information is also documented in the table.

Table 23. Compatibility of COPYTOCOPY with other utilities

<table>
<thead>
<tr>
<th>Action</th>
<th>Compatible with COPYTOCOPY?</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHECK DATA</td>
<td>Yes</td>
</tr>
<tr>
<td>CHECK INDEX</td>
<td>Yes</td>
</tr>
<tr>
<td>CHECK LOB</td>
<td>Yes</td>
</tr>
<tr>
<td>COPY</td>
<td>No</td>
</tr>
<tr>
<td>DIAGNOSE</td>
<td>Yes</td>
</tr>
<tr>
<td>LOAD</td>
<td>No</td>
</tr>
<tr>
<td>MERGECOPY</td>
<td>No</td>
</tr>
<tr>
<td>MODIFY</td>
<td>No</td>
</tr>
<tr>
<td>QUIESCE</td>
<td>Yes</td>
</tr>
<tr>
<td>REBUILD INDEX</td>
<td>Yes</td>
</tr>
<tr>
<td>RECOVER</td>
<td>No</td>
</tr>
<tr>
<td>REORG INDEX</td>
<td>No</td>
</tr>
<tr>
<td>REORG TABLESPACE</td>
<td>No</td>
</tr>
<tr>
<td>REPAIR</td>
<td>Yes</td>
</tr>
<tr>
<td>REPORT</td>
<td>Yes</td>
</tr>
<tr>
<td>RUNSTATS INDEX</td>
<td>Yes</td>
</tr>
<tr>
<td>RUNSTATS TABLESPACE</td>
<td>Yes</td>
</tr>
<tr>
<td>STOSPACE</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Table 23. Compatibility of COPYTOCOPY with other utilities  (continued)

<table>
<thead>
<tr>
<th>Action</th>
<th>Compatible with COPYTOCOPY?</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNLOAD</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Full or incremental image copies with COPYTOCOPY**

You can copy a full image copy or an incremental image copy by using FROMLASTCOPY keyword.

If you do not specify FROMLASTCOPY, it will be used by default, as shown in the following example. In this example, the COPYTOCOPY control statement specifies that the utility is to make a backup copy of the most recent full image copy or an incremental image copy of the table space DSN8S11E in database DSN8D11A:

COPYTOCOPY TABLESPACE DSN8D11A.DSN8S11E
    COPYDDN(,DDNAME2)

The COPYTOCOPY utility makes a copy from an existing image copy and writes pages from the image copy to the output data sets. The JCL for the utility job must include DD statements or a template for the output data sets. If the object consists of multiple data sets and all are copied in one job, the copies reside in one physical sequential output data set.

**Incremental image copies with COPYTOCOPY**

You can use COPYTOCOPY to make an incremental image copy. An incremental image copy is a copy of the pages that have changed since the last full or incremental image copy.

To make a copy of an incremental image copy, use the keyword FROMLASTINCRCOPY.

The following example control statement specifies that COPYTOCOPY is to make a local site backup image copy, a recovery site primary image copy, and a recovery site backup image copy from an incremental image copy.

COPYTOCOPY TABLESPACE DSN8D11A.DSN8S11E
    FROMLASTINCRCOPY
    COPYDDN(,COPY2)
    RECOVERYDDN(COPY3,COPY4)

**Using more than one COPYTOCOPY statement**

You can use more than one control statement for COPYTOCOPY in one Db2 utility job step.

**About this task**

After each COPYTOCOPY statement executes successfully:
- A row referring to the image copy is recorded in SYSIBM.SYSCOPY table.
- The image copy data set is valid and available for RECOVER, MERGECOPY, COPYTOCOPY, and UNLOAD.

If a job step that contains more than one COPYTOCOPY statement abnormally terminates, do not use TERM UTILITY. Restart the job from the last commit point.
by using RESTAR instead. Terminating COPYTOCOPY in this case might cause inconsistencies between the ICF catalog and Db2 catalogs if generation data sets are used.

### Copying from a specific image copy

You can specify a particular image copy that is to be used as input to the COPYTOCOPY utility by using the FROMCOPY option.

#### Procedure

Invoke the COPYTOCOPY utility with the FROMCOPY keyword. If the input data set is a FlashCopy image copy and the copied object is partitioned, you must also specify the data set number by including the DSNUM option in the control statement. If you specify the FROMCOPY keyword and the specified data set is not found in SYSIBM.SYSCOPY, COPYTOCOPY issues message DSNU1401I. Processing for the object then terminates.

#### Example

The following control statement specifies that COPYTOCOPY is to make three copies of the table space TPA9031C in database DBA90301 from the image copy data set DH109003.COPY1.STEP1.COPY3:

```plaintext
COPYTOCOPY TABLESPACE DBA90301.TPA9031C FROMCOPY DH109003.COPY1.STEP1.COPY3
COPYDDN(COPY2)
RECOVERYDDN(COPY3,COPY4)
```

### Copying a FlashCopy image copy by using COPYTOCOPY

COPYTOCOPY can create up to four sequential image copies of a FlashCopy image copy. COPYTOCOPY and COPY are the only utilities that can create sequential copies from a FlashCopy image copy.

#### About this task

Making sequential copies of a FlashCopy image copy is useful if you need to unload data from an image copy. Because the UNLOAD utility does not accept FlashCopy image copies as input, you must first make a sequential copy of the FlashCopy image copy and then unload the data from the sequential copy.

#### Procedure

To copy a FlashCopy image copy:

In the COPYTOCOPY utility control statement, specify all of the following options:
- FROMCOPY option with the name of VSAM data set that contains the FlashCopy image copy
- DSNUM option with the appropriate data set or partition number for the FlashCopy image copy
- COPYDDN and RECOVERYDDN as needed to indicate which sequential copies you want to create

#### Related concepts:

- “FlashCopy image copies” on page 157

#### Related tasks:
Using TEMPLATE with COPYTOCOPY

Template data set name substitution variables resolve as usual. COPYTOCOPY does not use the template values of the original COPY utility execution.

SYSCOPY records that are updated by COPYTOCOPY

The image copies COPYTOCOPY made are registered in SYSIBM.SYSCOPY for later use by the RECOVER utility. Other utilities can use these copies, too.

Columns that are inserted by COPYTOCOPY are the same as those of the original entries in SYSCOPY row when the COPY utility recorded them. Except for columns GROUP_MEMBER, JOBNAME, AUTHID, DSNAME, DEVTYPE, and DSVOLSER, the columns are those of the COPYTOCOPY job. When COPYTOCOPY is invoked at the partition level (DSNUM n) and the input data set is an inline copy that was created by the REORG of a range of partitions, COPYTOCOPY inserts zeros in the HIGHDSNUM and LOWDSNUM columns of the SYSCOPY record.

How COPYTOCOPY determines which input copy to use

The COPYTOCOPY utility makes a copy of an existing image copy. Which image copy the utility uses as input depends on the options that you specify and where you run the utility job.

If you specify the FROMCOPY keyword in the utility control statement, only the specified data set is used as input to the COPYTOCOPY job.

If you do not specify the FROMCOPY keyword, COPYTOCOPY uses the following search order to determine the input data set:

- If you run the utility at the local site, the search order is the local site primary copy, the local site backup copy, the recovery site primary copy, and the recovery site backup copy.
- If you run the utility at the recovery site, the search order is the recovery site primary copy, the recovery site backup copy, the local site primary copy, and the local site backup copy.

If the input data set cannot be allocated or opened, COPYTOCOPY attempts to use the next image copy data set, with the same START_RBA value in the SYSIBM.SYSCOPY catalog table, in the preceding search order.

Related reference:

- “Syntax and options of the COPYTOCOPY control statement” on page 186
- SYSIBM.SYSCOPY table (Db2 SQL)

Generation data group definitions for the COPYTOCOPY utility

You can use generation data groups to hold image copies. Use of generation data groups offers the benefit of automating allocation of data set names and deletion of the oldest data set. You can also use templates when using generation data groups.

To define the generation group, follow these guidelines:
• Use generation data groups to hold image copies because their use automates the allocation of data set names and the deletion of the oldest data set.
• Use templates when using generation data groups.
• When you define the generation data group:
  – You can specify that the oldest data set is to be automatically deleted when the maximum number of data sets is reached. If you do that, make the maximum number large enough to accommodate all recovery requirements. When data sets are deleted, use the MODIFY utility to delete the corresponding rows in SYSIBM.SYSCOPY.
  – Make the limit number of generation data sets equal to the number of copies that you want to keep. Use NOEMPTY to avoid deleting all the data sets from the integrated catalog facility catalog when the limit is reached.

### Using Db2 with DFSMS products

You can use Db2 with DFSMS products.

If image copy data sets are managed by HSM or SMS, all data sets are cataloged.

If you plan to use SMS, catalog all image copies. Never maintain cataloged and uncataloged image copies that have the same name.

### Image copies on tape

Do not combine a full image copy and incremental image copies for the same table space on one tape volume. If you do, the RECOVER TABLESPACE utility cannot allocate the incremental image copies.

### Copies of lists of objects from tape

The COPYTOCOPY utility determines the number of tape drives to use for the function.

If you use JCL to define tape drives, the JCL allocates tape drives for those definitions. If you use TEMPLATES to allocate tape drives for the output data sets, the utility dynamically allocates the tape drives according to the following algorithm:

• One tape drive if the input data set resides on tape.
• A tape drive for each template with STACK YES that references tape.
• Three tape drives, one for each of the local and remote output image copies, in case non-stacked templates reference tape.

Thus, COPYTOCOPY allocates a minimum of three tape drives. The utility allocates four tape drives if the input data set resides on tape, and more tape drives if you specified tape templates with STACK YES.

If input data sets to be copied are stacked on tape and output data sets are defined by a template, the utility sorts the list of objects by the file sequence numbers (FSN) of the input data sets and processes the objects serially.

Image copies of compressed indexes are copied in uncompressed format, so if you perform COPYTOCOPY using those image copies as input, it will result in uncompressed image copies.
For example, image copies of the following table spaces with their FSNs are stacked on TAPE1:
- Db2.TS1 FSN=1
- Db2.TS2 FSN=2
- Db2.TS3 FSN=3
- Db2.TS4 FSN=4

In the following statements, COPYTOCOPY uses a template for the output data set:

```plaintext
//COPYTOCOPY EXEC DSNUPROC,SYSTEM=V71A
//SYSIN DD *
TEMPLATE A1 &DB..&SP..COPY1 TAPE UNIT CART STACK YES
COPYTOCOPY
TABLESPACE DB1.TS4
   LASTFULL
   RECOVERYDDN(A1)
TABLESPACE DB1.TS1
   LASTFULL
   RECOVERYDDN(A1)
TABLESPACE DB1.TS2
   LASTFULL
   RECOVERYDDN(A1)
TABLESPACE DB1.TS3
   LASTFULL
   RECOVERYDDN(A1)
```

As a result, the utility sorts the objects by FSN and processes them in the following order:
- DB1.TS1
- DB1.TS2
- DB1.TS3
- DB1.TS4

If the output data sets are defined by JCL, the utility gives stacking preference to the output data sets over input data sets. If the input data sets are not stacked, the utility sorts the objects by size in descending order.

**Termination or restart of COPYTOCOPY**

You can terminate or restart the COPYTOCOPY utility.

**Termination of COPYTOCOPY**

You can use the TERM UTILITY command to terminate a COPYTOCOPY job.

**Restart of a COPYTOCOPY job**

If you do not use the TERM UTILITY command, you can restart a COPYTOCOPY job. COPYTOCOPY jobs restart from the last commit point. You cannot use RESTART(PHASE) for any COPYTOCOPY job. If you are restarting a COPYTOCOPY job with uncataloged output data sets, you must specify the appropriate volumes for the job in the JCL or on the TEMPLATE utility statement. Doing so could impact your ability to use implicit restart.

To prepare for restarting a COPYTOCOPY job, specify DISP=(MOD,CATLG,CATLG) on your DD statements.
Restart of COPYTOCOPY after an out-of-space condition

You can restart COPYTOCOPY from the last commit point after receiving an out-of-space condition.

Related tasks:
- “Terminating an online utility” on page 43
- “Restarting an online utility” on page 44
- “Restarting a utility after the output data set is full” on page 48

Sample COPYTOCOPY control statements

Use the sample control statements as models for developing your own COPYTOCOPY control statements.

Example 1: Making a local backup copy

The following control statement specifies that the COPYTOCOPY utility is to make a local backup copy of the most recent full image copy or incremental image copy, whichever is most recent. The COPYDDN option specifies that the data set for the local site backup image copy is defined by the COPY2 DD statement. Because no data set is specified for the local site primary image copy, which is usually the first parameter of the COPYDDN option, COPYTOCOPY expects this copy to already exist. If it does not exist, Db2 issues an error message and terminates the job.

```
//STEP1 EXEC DSNUPROC,UID='DH109001.COPY1',
  //   UTPROC='',
  //   SYSTEM='DSN'
//COPY2 DD DSN=DH109001.C2C01.STEP2.COPY2,DISP=(MOD,CATLG,CATLG),
  //   SPACE=(1000,(20,20),,,ROUND)
//SYSIN DD *
  COPYTOCOPY TABLESPACE DBA90101.TLA9011A COPYDDN(COPY2)
```

Example 2: Copying the most recent copy

The following control statement specifies that COPYTOCOPY is to make a local site backup copy, a recovery site primary copy, and a recovery site backup copy of table space DBA90102.TPA9012C. The COPYDDN and RECOVERYDDN options also indicate the data sets to which these copies should be written. For example, the recovery site primary copy is to be written to the COPY3 data set. The FROMLASTCOPY option specifies that the most recent full image copy or incremental image copy is to be used as the input copy data set. This option is the default and is therefore not required.

```
COPYTOCOPY TABLESPACE DBA90102.TPA9012C
  FROMLASTCOPY COPYDDN(COPY2)
  RECOVERYDDN(COPY3,COPY4)
```

Example 3: Copying the most recent full image copy

The following control statement specifies that COPYTOCOPY is to make primary and backup copies at the recovery site of table space DBA90201.TPA9021C. The FROMLASTFULLCOPY option specifies that the most recent full image copy is to be used as the input copy data set.

```
COPYTOCOPY TABLESPACE DBA90201.TPA9021C
  FROMLASTFULLCOPY
  RECOVERYDDN(COPY3,COPY4)
```
Example 4: Specifying a copy data set for input

The following control statement specifies that COPYTOCOPY is to make a local site backup copy, a recovery site primary copy, and a recovery site backup copy from data set DH109003.COPY1.STEP1.COPY3. This input data set is specified by the FROMCOPY option. The output data sets (COPY2, COPY3, and COPY4) are specified by the COPYDDN and RECOVERYDDN options.

```
COPYTOCOPY TABLESPACE DBA90301.TPA9031C
   FROMCOPY DH109003.COPY1.STEP1.COPY3
   COPYDDN(,COPY2)
   RECOVERYDDN(COPY3,COPY4)
```

Example 5: Identifying a cataloged image copy data set

The following control statement specifies that COPYTOCOPY is to make a local site backup copy from a cataloged data set that is named DH109003.COPY1.STEP1.COPY4. This data set is identified by the FROMCOPY and FROMVOLUME options. The FROMCOPY option specifies the input data set name, and the FROMVOLUME CATALOG option indicates that the input data set is cataloged. Use the FROMVOLUME option to distinguish a data set from other data sets that have the same name.

```
COPYTOCOPY TABLESPACE DBA90302.TLA9032A
   FROMCOPY DH109003.COPY1.STEP1.COPY4
   FROMVOLUME CATALOG
   COPYDDN(,COPY2)
```

Example 6: Identifying an uncataloged image copy data set

The control statement specifies that COPYTOCOPY is to make a local site backup copy, a recovery site primary copy, and a recovery site backup copy from an uncataloged data set, JUKQU2BP.COPY1.STEP1.TP01. The FROMCOPY option identifies this input data set name, and the FROMVOLUME option identifies the volume (SCR03) for the input data set. Use the FROMVOLUME option to distinguish a data set from other data sets that have the same name. The COPYDDN option identifies the data set for the local site backup copy. This data set is to be dynamically allocated according to the specifications of the C2C1_T1 template, which is defined in one of the preceding TEMPLATE control statements. The RECOVERYDDN option identifies the data sets for the recovery site copies. These data sets are to be dynamically allocated according to the specifications of the C2C1_T2 and C2C1_T3 templates, which are defined in the preceding TEMPLATE control statements.
Example 7: Processing a list of objects

The following control statement specifies that COPYTOCOPY is to make local site backup copies of the three partitions of table space DBA90402.TPA9042C that are specified by the DSNUM option (partitions 2, 3, and 4). COPYTOCOPY uses the following input copy data sets, as indicated by the FROMLASTFULLCOPY, FROMLASTCOPY, and FROMLASTINCRCOPY options:

- The most recent full image copy for partition 2
- The most recent full image copy or incremental image copy, whichever is most recent, for partition 3
- The most recent incremental image copy for partition 4

The COPYDDN option for each partition indicates the output data sets (COPY2, COPY3, and COPY4).

```
COPYTOCOPY TABLESPACE DBA90402.TPA9042C DSNUM 2
FROMLASTFULLCOPY COPYDDN(COPY2)
FROMLASTCOPY COPYDDN(COPY3)
FROMLASTINCRCOPY COPYDDN(COPY4)
```

Example 8: Using LISTDEF and TEMPLATE switching

The following COPYTOCOPY control statement specifies that the utility is to copy the list of objects that are included in the CPY1 list, which is defined by the LISTDEF control statement. The copies are to be written to the data sets that are defined by the T3 template, which is defined in the TEMPLATE control statement. Additionally, T3 template has defined the LIMIT keyword, that is to switch from T3 template to T4 template if the output data set size is bigger than the specified limit value 5 MB. This template defines the naming convention for the output data sets that are to be dynamically allocated.
The OPTIONS PREVIEW statement before the LISTDEF statement is used to force the CPY1 list contents to be included in the output. For long lists, using this statement is not recommended, because it might cause the output to be too long. The OPTIONS OFF statement ends the PREVIEW mode processing, so that the following TEMPLATE and COPYTOCOPY jobs run normally.

```plaintext
OPTIONS PREVIEW
LISTDEF CPY1 INCLUDE TABLESPACES TABLESPACE DBA906*.T*A906*
       INCLUDE INDEXSPACES INDEXSPACE ADMF001.I?A906*
OPTIONS OFF
TEMPLATE T4 UNIT(380)
       DSN(T4.&SN..T&TI..COPY&IC..&LOCREM.)
TEMPLATE T3 UNIT(SYSDA) SPACE CYL
       DSN(T3.&SN..T&TI..COPY&IC..&LOCREM.)
       LIMIT(5 MB,T4)
COPYTOCOPY LIST CPY1 COPYDDN(T3,T3)
```

**Example 8: Using LISTDEF and TEMPLATE with the CLONE option**

The following COPYTOCOPY control statement specifies that the utility is to copy the list of objects that are included in the C2C1_LIST list, which is defined by the LISTDEF control statement. The CLONE option indicates that COPYTOCOPY is to process only image copy data sets that were taken against clone objects.

```plaintext
LISTDEF C2C1_LIST
       INCLUDE TABLESPACES TABLESPACE DBKQBS01.TPKQBS01
       INCLUDE INDEXSPACES INDEXSPACE DBKQBS01.IPKQBS11
       INCLUDE INDEXSPACES INDEXSPACE DBKQBS01.IXKQBS12
       INCLUDE TABLESPACES TABLESPACE DBKQBS02.TSKQBS02
       INCLUDE INDEXSPACES INDEXSPACE DBKQBS02.IXKQBS21
       INCLUDE INDEXSPACES INDEXSPACE DBKQBS02.IXKQBS22

TEMPLATE C2C1_T1
       DSN(JUKQU2BS.C2C1.LB.&SN.)
       DISP(NEW,CATLG,CATLG)
       UNIT(SYSDA)

TEMPLATE C2C1_T2
       DSN(JUKQU2BS.C2C1.RP.&SN.)
       DISP(NEW,CATLG,CATLG)
       UNIT(SYSDA)

TEMPLATE C2C1_T3
       DSN(JUKQU2BS.C2C1.RB.&SN.)
       DISP(NEW,CATLG,CATLG)
       UNIT(SYSDA)

COPYTOCOPY LIST C2C1_LIST
       FROMLASTFULLCOPY
       COPYDDN(C2C1_T1)
       RECOVERYDDN(C2C1_T2,C2C1_T3)
       CLONE
```
Chapter 16. DIAGNOSE

The DIAGNOSE online utility generates information that is useful in diagnosing problems. Use this utility only under the direction of IBM Software Support.

At the direction of IBM Support, you can use the DIAGNOSE utility to complete the following tasks:

- Obtain dumps for any utility abend, overriding any considerations that might suppress such a dump.
- Send OBD and SYSUTIL information to SYSPRINT for review (DISPLAY option).
- Dump both the Db2 MEPL and the application space (DSNUTILB) MEPL to SYSPRINT without forcing an SVC dump (DISPLAY MEPL option).
- Dump an entry for a database in SYSIBM.SYSDATABASE to SYSPRINT (DISPLAY option).
- Force Db2 to abend after a specific Db2 message is specified or when a module trace ID is encountered (ABEND option with the MESSAGE or TRACEID option).

Interpreting output

One intended use of this utility is to aid in determining and correcting system problems. When diagnosing Db2 problems, you might need to refer to licensed documentation to interpret output from this utility.

Authorization required

To execute this utility for options which access relational data, you must use a privilege set that includes one of the following authorizations:

- REPAIR privilege for the database
- DBADM or DBCTRL authority for the database. If the object on which the utility operates is in an implicitly created database, DBADM authority on the implicitly created database or DSNDB04 is required.
- System DBADM authority
- DATAACCESS authority
- SQLADM authority
- SYSCTRL or SYSADM authority

An ID with installation SYSOPR authority can execute the DIAGNOSE utility on a table space in the DSNDB01 or DSNDB06 database.

An ID with installation SYSADM authority can execute the DIAGNOSE utility with the WAIT statement option on any table space.

You can create a control statement with the ISPF/PDF edit function. After you create it, save it in a sequential or partitioned data set. When you create the JCL for running the job, use the SYSIN DD statement to specify the name of the data set that contains the utility control statement.
Syntax diagram

diagnose statement:

- TYPE(integer)
- ALLDUMPS
- (X'abend-code')
- NODUMPS
- (X'abend-code')

display statement:

- DISPLAY OBD database-name..table-space-name ALL CLONE
- TABLES
- INDEXES
- SYSUTIL
- MEPL
- AVAILABLE
- OBET DATABASE database-name TABLESPACE database-name..table-space-name CLONE
- INDEX index-name

wait statement:
Option descriptions

**TYPE(integer, ...)**
Specifies one or more types of diagnosis that you want to perform.

*integer* is the number of types of diagnoses. The maximum number of types is 32. IBM Support defines the types as needed to diagnose problems with IBM utilities.

**ALLDUMPS(X'abend-code', ...)**
Forces a dump to be taken in response to any utility abend code.

*X'abend-code'* is a member of a list of abend codes to which the scope of ALLDUMPS is limited.

*abend-code* is a hexadecimal value.

**NODUMPS(X'abend-code', ...)**
Suppresses the dump for any utility abend code.

*X'abend-code'* is a member of a list of abend codes to which the scope of NODUMPS is limited.

*abend-code* is a hexadecimal value.

**DISPLAY**
Formats the specified database items using SYSPRINT.

**OBD database-name.table-space-name**
Formats the object descriptor (OBD) of the table space.

*database-name* is the name of the database in which the table space belongs.

*table-space-name* is the name of the table space whose OBD is to be formatted.

**ALL**
Formats all OBDs of the table space. The OBD of any object that is associated with the table space is also formatted.

**TABLES**
Formats the OBDs of all tables in the specified table spaces.
INDEXES
 Formats the OBDs of all indexes in the specified table spaces.

SYSUTIL
 Formats every record from SYSIBM.SYSUTIL. This directory table stores information about all utility jobs.

MEPL
 Dumps the module entry point lists (MEPLs) to SYSPRINT.

AVAILABLE
 Displays the utilities that are installed on this subsystem in both bitmap and readable format. The presence or absence of the Db2 Utilities Suite for z/OS (5655-W87) affects the results of this display. Message DSNU862I displays the output from DIAGNOSE DISPLAY AVAILABLE.

Related information:
 DSNU862I (Db2 Messages)

DBET
 Dumps the contents of a database exception table (DBET) to SYSPRINT.

DATABASE database-name
 Dumps the DBET entry that is associated with the specified database.

   database-name is the name of the database.

TABLESPACE database-name.table-space-name
 Dumps the DBET entry that is associated with the specified table space.

   database-name is the name of the database.

   table-space-name is the name of the table space.

INDEX creator-name.index-name
 Dumps the DBET entry that is associated with the specified index.

   creator-name is the ID of the creator of the index.

   index-name is the name of the index.

   Enclose the index name in quotation marks if the name contains a blank.

CLONE
 Indicates that DIAGNOSE is to display information for only the following specified objects:

   • Clone tables
   • Table spaces that contain clone tables
   • Indexes on clone tables
   • Index spaces that contain indexes on clone tables.

WAIT
 Suspends utility execution when it encounters the specified utility message or utility trace ID. DIAGNOSE issues a message to the console. Utility execution does not resume until the operator replies to that message, the utility job times out, or the utility job is canceled. This waiting period allows events to be synchronized while you are diagnosing concurrency problems. The utility waits for the operator to reply to the message, allowing the opportunity to time or synchronize events.

   If the utility message or trace ID is not encountered, processing continues.
ABEND
Forces an abend during utility execution if the specified utility message or utility trace ID is issued.

If the utility message or trace ID is not encountered, processing continues

NODUMP
Suppresses the dump that is generated by an abend of DIAGNOSE.

MESSAGE message-id
Specifies a DSNUxxx or DSNUxxxx message that causes a wait or an abend to occur when that message is issued.

message-id is the message, in the form of Uxxx or Uxxxx.

INSTANCE integer
Specifies that a wait or an abend is to occur when the MESSAGE option message is encountered a specified number of times. If INSTANCE is not specified, a wait or abend occurs each time that the message is encountered.

integer is the number of times that a message is to be encountered before a wait or an abend occurs.

TRACEID trace-id
Specifies a trace ID that causes a wait or an abend to occur when the ID is encountered. You can find valid trace IDs can be found in data set prefix.SDSNSAMP(DSNWEIDS).

trace-id is a trace ID that is associated with the utility trace (RMI21). You can specify trace-id in either decimal (integer) or hexadecimal (X'trace-id') format.

INSTANCE integer
Specifies that a wait or an abend is to occur when the TRACEID option is encountered a specified number of times. If INSTANCE is not specified, a wait or abend occurs each time that the trace ID is encountered.

integer is the number of times that a trace ID is to be encountered before a wait or an abend occurs.

END
Ends DIAGNOSE processing.

Data sets that DIAGNOSE uses

The following table lists the data sets that DIAGNOSE uses. The table lists the DD name that is used to identify the data set, a description of the data set, and an indication of whether it is required. Include statements in your JCL for each required data set.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSIN</td>
<td>Input data set that contains the utility control statement.</td>
<td>Yes</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>Output data set for messages.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

The following objects are named in the utility control statement and do not require DD statements in the JCL:

Database
Database about which DIAGNOSE is to gather diagnosis information.
Table space
Table space about which DIAGNOSE is to gather diagnosis information.

Index space
Index about which DIAGNOSE is to gather diagnosis information.

Concurrency and compatibility
DIAGNOSE can run concurrently on the same target object with any SQL operation or utility, except a utility that is running on DSNDB01.SYSUTILX.

Termination or restart
You can terminate a DIAGNOSE utility job by using the TERM UTILITY command if you submitted the job or have SYSOPR, SYSCTRL, or SYSADM authority.

You can restart a DIAGNOSE utility job, but it starts from the beginning again.

Example 1: Displaying Db2 MEPLs
The following DIAGNOSE utility control statement specifies that the Db2 MEPLs are to be displayed. You can use the output from this statement to find the service level of a specific Db2 module. The output lists each module, the most recent PTF or APAR that was applied to the module, and the date that the PTF or APAR was installed.

Important: Specify Db2 load libraries in the JOBLIB or STEPLIB of the DIAGNOSE DISPLAY MEPL job that are at the same maintenance level as the load libraries for the Db2 subsystem on which you execute the utility. Doing this ensures that the information that DIAGNOSE DISPLAY MEPL reports reflects the current state of the Db2 subsystem.

DIAGNOSE
DISPLAY MEPL

Example 2: Forcing a dump
The following control statement forces a dump if an abend occurs with either of the following reason codes: X'00E40322' or X'00E40323'.

DIAGNOSE
ALLDUMPS(X'00E40322',X'00E40323')

The following control statement forces a dump for any utility abend that occurs during the execution of the specified COPY job. The DIAGNOSE END option ends DIAGNOSE processing.

DIAGNOSE
ALLDUMPS
COPY TABLESPACE DSNDB06.SYSDDF
DIAGNOSE END

Example 3: Performing a diagnosis of a specific type
The control statement in this example specifies that you want to perform a diagnosis of type 66. Run this job under the direction of IBM Support to diagnose problems with utility parallelism.
Example 4: Forcing a utility abend

The control statement in this example forces an abend of the specified COPY job when one instance of message DSNU400 is issued. The NODUMP option indicates that the DIAGNOSE utility is not to generate a dump in this situation.

```
//STEP3 EXEC DSNUPROC,UID='JUOSU226.REBUI',
   // UTPROC='',SYSTEM='SSTR'
//SYSIN DD *
DIAGNOSE TYPE(66)
    REBUILD INDEX (IDOS0302, IDOS0304, IPOS0301)
    SORTDEVT SYSDA SORTNUM 3
DIAGNOSE END
/*
Figure 21. Example of diagnosing type 66

Example 5: Suspending utility execution

The control statement in this example indicates that the specified COPYTOCOPY job is to be suspended when it encounters 51 occurrences of the trace ID X'2E6F'.

```
//STEP1 EXEC DSNUPROC,UID='IUJMUI16.COPY1',
   // UTPROC='',
   // SYSTEM='DSN'
//SYSCOPY1 DD DSN=IUJMUI16.COPY.STEP1.SYSCOPY1,DISP=(NEW,CATLG,CATLG),
   // UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSIN DD *
DIAGNOSE ABEND MESSAGE U400
    INSTANCE 1
    NODUMP
    COPY TABLESPACE DSN8D11A.DSN8S11E
    COPYDDN SYSCOPY1
DIAGNOSE END
/*

The following control statement forces an abend of the specified LOAD job when message DSNU311 is issued for the fifth time. The NODUMP option indicates that the DIAGNOSE utility is not to generate a dump in this situation.

```
DIAGNOSE
   ABEND MESSAGE U311 INSTANCE 5 NODUMP
LOAD DATA RESUME NO
   INTO TABLE TABLE1
     (NAME POSITION(1) CHAR(20))
DIAGNOSE END
/*
Figure 22. Example of forcing an abend of the COPY utility

Example 5: Suspending utility execution

The control statement in this example indicates that the specified COPYTOCOPY job is to be suspended when it encounters 51 occurrences of the trace ID X'2E6F'.
Example 6: Displaying only CLONE data

The control statement indicates that the DIAGNOSE utility is to display information for only the specified objects that are table clones, table spaces that contain clone tables, indexes on clone tables, or index spaces that contain indexes on clone tables.

DIAGNOSE DISPLAY DBET
  DATABASE DBNI0501
  CLONE

Related concepts:

Db2 utilities for troubleshooting (Diagnosing Db2 problems)

How to force a utility abend

You can force a utility abend by specifying either a message or a trace IFCID in the DIAGNOSE utility control statement.

Procedure

To force utilities to abend, use the following approaches:

- Issue a DIAGNOSE utility control statement. To force an abend when unique-index or referential-constraint violations are detected, you must specify the message that is issued when the error is encountered. Specify this message by using the MESSAGE option of the ABEND statement.
- Instead of using a message, you can force an abend by using the TRACEID option of the ABEND statement to specify a trace IFCID that is associated with the utility to force an abend.
- Use the INSTANCE keyword to specify the number of times that the specified message or trace record is to be generated before the utility abends.
Chapter 17. EXEC SQL

The EXEC SQL online utility control statement declares cursors or executes dynamic SQL statements. You can use this utility as part of the Db2 cross-loader function of the LOAD utility.

The cross-loader function enables you to use a single LOAD job to transfer data from one location to another location or from one table to another table at the same location. You can use either a local server or any DRDA-compliant remote server as a data input source for populating your tables. Your input can even come from other sources besides Db2 for z/OS; you can use IBM Information Integrator Federation feature for access to data from sources as diverse as Oracle and Sybase, as well as the entire Db2 family of database servers.

Output

The EXEC SQL control statement produces a result table when you specify a cursor.

Authorization required

The EXEC SQL statement itself requires no privileges to execute. The authorization rules that are defined for the dynamic preparation of the SQL statement specified by EXECUTE IMMEDIATE apply.

Execution phases of EXEC SQL

The EXEC SQL control statement executes entirely in the EXEC phase. You can restart the EXEC phase if necessary.

Concurrency and compatibility of EXEC SQL

You can use the EXEC SQL control statement with any utility that allows concurrent SQL access on a table space. Other databases are not affected.

EXEC SQL syntax

You can create a control statement with the ISPF/PDF edit function. After creating it, save it in a sequential or partitioned data set. When you create the JCL for running the job, use the SYSIN DD statement to specify the name of the data set that contains the utility control statement.

Utility control statements submitted in UNICODE, including EXEC SQL, are translated into EBCDIC before processing; however, character string constants are not translated. Character string constants are left in the character set in which they were specified. In some cases, you might need to use hexadecimal string constants in order to achieve the behavior that you want.
declare-cursor-spec:

```
DECLARE cursor-name CURSOR FOR select-statement
```

Option descriptions

cursor-name
- Specifies the cursor name. The name must not identify a cursor that is already declared within the same input stream. When using the Db2 cross-loader function to load data from a remote server, you must identify the cursor with a three-part name. Cursor names that are specified with the EXEC SQL utility **cannot be longer than eight characters**.

select-statement
- Specifies the result table for the cursor. This statement can be any valid SQL SELECT statement, including joins, unions, conversions, aggregations, special registers, and user-defined functions.

non-select dynamic SQL statement
- Specifies a dynamic SQL statement that is to be used as input to EXECUTE IMMEDIATE. You can specify the following dynamic SQL statements in a utility statement:
  - ALTER
  - COMMENT ON
  - COMMIT
  - CREATE
  - DELETE
  - DROP
  - EXPLAIN
  - GRANT
  - INSERT
  - LABEL ON
  - LOCK TABLE
  - RENAME
  - REVOKE
  - ROLLBACK
  - SET CURRENT DECFLOAT Rounding MODE
  - SET CURRENT DEGREE
  - SET CURRENT LOCALE LC_CTYPE
  - SET CURRENT OPTIMIZATION HINT
  - SET PATH
  - SET CURRENT PRECISION
  - SET CURRENT RULES
  - SET CURRENT SQLID
  - UPDATE

Each SQL statement runs as a separate thread. When the utility executes the SQL statement, the specified statement string is parsed and checked for errors.
If the SQL statement is invalid, EXEC SQL does not execute the statement and reports the error condition. If the SQL statement is valid, but an error occurs during execution, EXEC SQL reports that error condition. When an error occurs, the utility terminates.

**Termination or restart of EXEC SQL**

You can terminate an EXEC SQL utility job by using the TERM UTILITY command if you submitted the job or have SYSOPR, SYSCTRL, or SYSADM authority.

You can restart an EXEC SQL utility job, but it starts from the beginning again. If you are restarting this utility as part of a larger job in which EXEC SQL completed successfully, but a later utility failed, do not change the EXEC SQL utility control statement, if possible. If you must change the EXEC SQL utility control statement, use caution; any changes can cause the restart processing to fail.

**Example 1: Creating a table**

The following control statement specifies that Db2 is to create table MYEMP with the same rows and columns as sample table EMP.

```sql
EXEC SQL
   CREATE TABLE MYEMP LIKE DSN8B10.EMP CCSID EBCDIC
ENDEXEC
```

This type of statement can be used to create a mapping table.

**Example 2: Inserting rows into a table**

The following control statement specifies that Db2 is to insert all rows from sample table EMP into table MYEMP.

```sql
EXEC SQL
   INSERT INTO MYEMP SELECT * FROM DSN8B10.EMP
ENDEXEC
```

**Example 3: Declaring a cursor**

The following control statement declares C1 as the cursor for a query that is to return all rows from table DSN8810.EMP.

```sql
EXEC SQL
   DECLARE C1 CURSOR FOR SELECT * FROM DSN8810.EMP
ENDEXEC
```
You can use a declared cursor with the Db2 cross-loader function to load data from a local server or from any DRDA-compliant remote server as part of the Db2 cross-loader function.

**Related tasks:**
- “Loading data by using the cross-loader function” on page 333

**Related reference:**
- [Statements (Db2 SQL)](#)
- [select-statement (Db2 SQL)](#)
Chapter 18. LISTDEF

The LISTDEF utility enables you to group database objects into reusable lists. You can then specify these lists in other utility control statements to indicate that the utility is to process all of the items in the list.

You can use LISTDEF to standardize object lists and the utility control statements that refer to them. This standardization reduces the need to customize or alter utility job streams.

If you do not use lists and you want to run a utility on multiple objects, you must run the utility multiple times or specify an itemized list of objects in the utility control statement.

Output

Output from the LISTDEF control statement consists of a list with a name.

Authorization required

To execute the LISTDEF utility, you must have SELECT authority on SYSIBM.SYSINDEXES, SYSIBM.SYSTABLES, and SYSIBM.SYSTABLESPACE.

You must use a privilege set that includes one of the following authorities:
- SELECT authority on SYSIBM.SYSINDEXES, SYSIBM.SYSTABLES, and SYSIBM.SYSTABLESPACE
- SQLADM authority
- DATAACCESS authority
- System DBADM authority
- SYSCTRL or SYSADM authority

Additionally, you must have the authority to execute the utility that is used to process the list, as currently documented in the “Authorization required” topic for each utility.

If you do not have authorization to execute the utility on one or more of the items in the list, the utility will stop on the first authorization error. To skip items in the list that return an error, use the OPTIONS (ITEMERROR, SKIP) control statement.

Execution phases of LISTDEF

The LISTDEF control statement executes entirely within the UTILINIT phase.

Syntax and options of the LISTDEF control statement

The LISTDEF utility control statement, with its multiple options, defines a list of table spaces, index spaces, or both on which other utilities can operate.

You can create a control statement with the ISPF/PDF edit function. After you create it, save it in a sequential or partitioned data set. When you create the JCL...
for running the job, use the SYSIN DD statement to specify the name of the data set that contains the utility control statement.

**Syntax diagram**

```
LISTDEF list-name list-options:

list-options:

INCLUDE
EXCLUDE

(1)
type-spec

LIST-referenced-list

initial-object-spec

CLONED YES

NOT

(2)

DEFINED YES

DEFINED NO

DEFINED ALL

BASE
LOB
XML
ALL

HISTORY
ARCHIVE

BASIC YES

EXTENDED NO

NOT
```

**Notes:**

1. You must specify `type-spec` if you specify DATABASE.
2. HISTORY or ARCHIVE can be specified either before or after the BASE, LOB, XML, and ALL keywords. However, HISTORY and ARCHIVE are always processed last, after all other keywords are handled.

**type-spec:**

```
TABLESPACES
INDEXSPACES

COPY NO

YES
```

**initial-object-spec:**
Option descriptions

LISTDEF list-name
Defines a list of Db2 objects and assigns a name to the list. The list name makes the list available for subsequent execution as the object of a utility control statement or as an element of another LISTDEF statement.

list-name is the name (up to 18 alphanumeric characters in length) of the defined list.

list-name cannot be a utility keyword name.

You can put LISTDEF statements either in a separate LISTDEF library data set or before a Db2 utility control statement that references the list-name.

INCLUDE
Specifies that the list of objects that results from the expression that follows is to be added to the list. You must first specify an INCLUDE clause. You can then specify subsequent INCLUDE or EXCLUDE clauses in any order to add to or delete clauses from the existing list.

Referencing a large number of databases in a single LISTDEF list might fill the EDM DBD pool and result in an abend 04E with reason code 00C90089. If this situation occurs, decrease the number of databases that are referenced in the LISTDEF statement. If this condition occurs frequently, increase the size of your EDM DBD pool.

Related information:
Calculating EDM pool sizes (Db2 Installation and Migration)

EXCLUDE
Specifies, after the initial INCLUDE clause, a list of objects to exclude. The expression that follows the EXCLUDE keyword determines this list of objects to exclude. These objects are excluded from the existing LISTDEF list if the objects are already in the list. If the objects are not in the existing list, they are ignored, and Db2 proceeds to the next INCLUDE or EXCLUDE clause.

TABLESPACES
Specifies that the INCLUDE or EXCLUDE object expression is to create a list of related table spaces.
TABLESPACES is the default type for lists that use a table space or a table for the initial search. For more information about specifying these objects, see the descriptions of the TABLESPACE and TABLE options.

No default type value exists for lists that use other lists for the initial search. The list that is referenced in the LIST option is used unless you specify TABLESPACES or INDEXSPACES. Likewise, no type default value exists for lists that use databases for the initial search. If you specify the DATABASE option, you must specify INDEXSPACES or TABLESPACES. For more information about specifying lists and databases, see the descriptions of the LIST and DATABASE options.

The result of the TABLESPACES keyword varies depending on the type of object that you specify in the INCLUDE or EXCLUDE clause. These results are shown in the following table.

<table>
<thead>
<tr>
<th>Object type that is specified in INCLUDE or EXCLUDE clause</th>
<th>Result of the TABLESPACES keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATABASE</td>
<td>Returns all table spaces that are contained within the database</td>
</tr>
<tr>
<td>TABLESPACE</td>
<td>Returns the specified table space</td>
</tr>
<tr>
<td>TABLE</td>
<td>Returns the table space that contains the table</td>
</tr>
<tr>
<td>INDEXSPACE</td>
<td>Returns the table space that contains the related table</td>
</tr>
<tr>
<td>INDEX</td>
<td>Returns the table space that contains the related table</td>
</tr>
<tr>
<td>LIST of table spaces</td>
<td>Returns the table spaces from the expanded referenced list</td>
</tr>
<tr>
<td>LIST of index spaces</td>
<td>Returns the related table spaces for the index spaces in the expanded referenced list</td>
</tr>
<tr>
<td>LIST of table spaces and index spaces</td>
<td>Returns the table spaces from the expanded referenced list and the related table spaces for the index spaces in the same list</td>
</tr>
</tbody>
</table>

INDEXSPACES

Specifies that the INCLUDE or EXCLUDE object expression is to create a list of related index spaces.

INDEXSPACES is the default type for lists that use an index space or an index for the initial search. For more information about specifying these objects, see the descriptions of the INDEXSPACE and INDEX options.

No default type value exists for lists that use other lists for the initial search. The list that is referenced in the LIST option is used unless you specify TABLESPACES or INDEXSPACES. Likewise, no type default value exists for lists that use databases for the initial search. If you specify the DATABASE option, you must specify INDEXSPACES or TABLESPACES. For more information about specifying lists and databases, see the descriptions of the LIST and DATABASE options.

The result of the INDEXSPACES keyword varies depending on the type of object that you specify in the INCLUDE or EXCLUDE clause. These results are shown in the following table.
Table 26. Result of the INDEXSPACES keyword based on the object type that is specified in the INCLUDE or EXCLUDE clause.

<table>
<thead>
<tr>
<th>Object type that is specified in INCLUDE or EXCLUDE clause</th>
<th>Result of the INDEXSPACES keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATABASE</td>
<td>Returns all index spaces that are contained within the database</td>
</tr>
<tr>
<td>TABLESPACE</td>
<td>Returns all index spaces for indexes over all tables in the table space</td>
</tr>
<tr>
<td>TABLE</td>
<td>Returns all index spaces for indexes over the table</td>
</tr>
<tr>
<td>INDEXSPACE</td>
<td>Returns the specified index space.</td>
</tr>
<tr>
<td>INDEX</td>
<td>Returns the index space that contains the index</td>
</tr>
<tr>
<td>LIST of table spaces</td>
<td>Returns the related index spaces for the table spaces in the expanded referenced list</td>
</tr>
<tr>
<td>LIST of index spaces</td>
<td>Returns the index spaces from the expanded referenced list</td>
</tr>
<tr>
<td>LIST of table spaces and index spaces</td>
<td>Returns the index spaces from the expanded referenced list and the related index spaces for the table spaces in the same list</td>
</tr>
</tbody>
</table>

COPY

Specifies whether indexes with COPY YES or COPY NO attributes are to be included or excluded in this portion of the list. If you omit COPY, all index spaces that satisfy the INCLUDE or EXCLUDE expression, regardless of their COPY attribute, are included or excluded in this portion of the list. If specified, this keyword must immediately follow the INDEXSPACES keyword. If you specify this keyword elsewhere, it is interpreted as the start of the COPY utility control statement.

YES

Specifies that only index spaces that were defined with or altered to COPY YES are to be included in this portion of the list. Use INCLUDE with COPY YES to develop a list of index spaces that the COPY utility can process.

NO

Specifies that only index spaces that were defined with or altered to COPY NO are to be included in this portion of the list. Use EXCLUDE with COPY NO to remove indexes that the COPY utility cannot process from a larger list.

LIST referenced-list

Specifies the name of a previously defined object list that is to be expanded and used for the initial search for the object.

referenced-list is the name of the list. You must explicitly specify this name. You cannot specify pattern-matching characters (%, *, ?, and _) for lists.

No default type value exists for lists that are developed from the LIST option. The list is expanded as defined, and it is then modified by subsequent keywords, if any.

You can specify a type-spec of TABLESPACES to create a list of only table spaces. If the list to be processed contains index spaces, the TABLESPACES keyword creates a list that includes related table spaces.

You can specify a type-spec of INDEXSPACES to create a list of only index spaces. If the list to be processed contains table spaces, the INDEXSPACES keyword creates a list that includes related index spaces.

You can use the LIST keyword to perform any of the following actions:
• Make aggregate lists of lists
• Exclude entire lists from other lists
• Develop lists of objects that are related to other lists

The partitions or partition ranges can be specified in a list.

**DATABASE database-name**

Specifies the database that is to be used for the initial search for the object.

You can specify the `database-name` explicitly or as a pattern-matched name. `DATABASE *` and `DATABASE %` are not supported.

If you specify `DATABASE`, you must also specify either `TABLESPACES` or `INDEXSPACES` as the list type. Depending on the list type that you specify, Db2 includes all table spaces or index spaces in `database-name` that satisfy the pattern-matching expression in the list.

You cannot specify DSNDB01, DSNDB06, DSNDB07, or user-defined work file databases in a `LISTDEF`.

Use caution when you specify an implicit `DATABASE` name. Authorization to access objects that are within an implicit database is not uniform. Use the `OPTIONS EVENT (ITEMERROR, SKIP)` control statement to continue processing when authorization errors occur.

**TABLESPACE database-name.table-space-name**

Specifies the table space that is to be used for the initial search for the object.

If you specify `TABLESPACE`, the default list type is `TABLESPACES`. All table spaces that satisfy the pattern-matching expression are included in the list unless the list is modified by other keywords. `TABLESPACE *.*` and `TABLESPACE %.%` are not supported.

`database-name` specifies the name of the database to which the table space belongs. The default value is `DSNDB04`.

`table-space-name` specifies the name of the table space.

You can explicitly specify or use pattern-matching characters to specify `database-name`, `table-space-name`, or both.

You cannot include any objects in DSNDB07 or any user-defined work file databases in a `LISTDEF`. Pattern matching is not supported for DSNDB01 and DSNDB06 objects.

**INDEXSPACE database-name.index-space-name**

Specifies the index space that is to be used for the initial search for the object.

If you specify `INDEXSPACE`, the default list type is `INDEXSPACES`. All index spaces that satisfy the pattern-matching expression are included in the list unless the index spaces are excluded by other `LISTDEF` options. `INDEXSPACE *.*` and `INDEXSPACE %.%` are not supported.

`database-name` specifies the name of the database to which the index space belongs. The default value is `DSNDB04`.

`index-space-name` specifies the name of the index space.

You can explicitly specify or use pattern-matching characters to specify `database-name`, `index-space-name`, or both.

You cannot include any objects in DSNDB07 or any user-defined work file databases in a `LISTDEF`. Pattern matching is not supported for DSNDB01 and DSNDB06 objects.
TABLE  creator-id.table-name  
Specifies the table that is to be used for the initial search for the object.
If you specify TABLE, the default list type is TABLESPACES. All table spaces that contain tables that satisfy the pattern-matching expression are included in the list unless the list is modified by other keywords. TABLE *.* and TABLE %.% are not supported.

creator-id specifies the qualifier creator ID for the table. The default value is the user identifier for the utility. table-name specifies the name of the table. If you specify a table name with CLONED, the CLONED keyword is ignored.
You can explicitly specify or use pattern-matching characters to specify creator-id,table-name, or both. However, the underscore pattern-matching character is ignored in a table name.
Pattern-matching is not supported for catalog and directory objects. In a LISTDEF statement, you must include catalog and directory objects by their fully qualified names.
Enclose the table name in quotation marks if the name contains a blank.

INDEX creator-id.index-name  
Specifies the index that is to be used for the initial search for the object.
If you specify INDEX, the default list type is INDEXSPACES. All index spaces that contain indexes that satisfy the pattern-matching expression are included in the list unless the list is modified by other keywords. INDEX *.* and INDEX %.% are not supported.

creator-id specifies the qualifier creator ID for the index. The default value is the user identifier for the utility.
index-name specifies the name of the index.
Enclose the index name in quotation marks if the name contains a blank.
You can explicitly specify or use pattern-matching characters to specify creator-id, index-name, or both. However, the underscore pattern-matching character is ignored in an index name.
Pattern-matching is not supported for catalog and directory objects. In a LISTDEF statement, you must include catalog and directory objects by their fully qualified names.

PARTLEVEL  
Specifies the partition granularity for partitioned table spaces, partitioning indexes, and data-partitioned secondary indexes that are to be contained in the list. You cannot specify the PARTLEVEL keyword with the RI keyword.

(integer)  
integer is the physical partition number where integer >= 0.

If you specify PARTLEVEL 0, the resulting list contains one entry for each nonpartitioned object.

If you specify PARTLEVEL with a nonzero operand, the resulting list contains one entry for the specified partition for partitioned objects and one entry for each nonpartitioned object.

If you specify PARTLEVEL without (integer), the resulting list contains one entry for each partition in the partitioned object and one entry for each nonpartitioned object.
Designates a range of partitions to be specified in the list. `integer1` and `integer2` are physical partition numbers and must be greater than 1. You can specify the range in any order. For example, you can specify (100:199) or (199:100). In either case, partitions 100 through 199 are specified in the list. If `integer1` is equal to `integer2`, only that one partition is specified. For example, if you specify (5:5), only partition 5 is specified.

An INCLUDE with the PARTLEVEL keyword can be removed from the list only by an EXCLUDE with PARTLEVEL.

For partition-by-growth objects, the PARTLEVEL keyword results in an entry for each partition that exists when the LISTDEF list is evaluated. Partitions that are added after the list is evaluated are not included in the list. If a partition is added during long-running job steps in which the list is reused, the partitions that were added are not included in the list and not processed. If a utility job that uses a PARTLEVEL list is restarted, the original list is saved during the original execution for a later restart. The list does not include any added partitions.

**CLONED**

Use the CLONED keyword to have LISTDEF perform a final filtering of the INCLUDE or EXCLUDE clause contents based on the existence or absence of clone data. This operation is performed last, after LISTDEF processes all other keywords on the INCLUDE or EXCLUDE clause.

CLONED YES specifies that only table spaces and index spaces that contain cloned objects are to be returned in the INCLUDE or EXCLUDE clause.

CLONED NO specifies that only table spaces and index spaces that do not contain cloned objects are to be returned in the INCLUDE or EXCLUDE clause.

Omit the CLONED keyword if the existence of clone data is not a factor.

The use of CLONED YES or CLONED NO affects only the contents of the list. It does not determine whether clone or base data is later processed by the utility that uses the list. Only the presence or absence of the CLONE keyword on individual utility control statements determines whether clone or base data is processed.

**DEFINED**

Specifies whether table spaces or index spaces with defined or undefined data sets are to be returned in the INCLUDED or EXCLUDED clause. If you omit the DEFINED keyword, DEFINED YES is the default.

**YES**

Specifies that only table spaces or index spaces that are currently defined are to be included in the INCLUDED or EXCLUDED clause.

YES is the default if DEFINED is not specified. By default, only defined objects are included in the list. Before DB2 10, the DEFINED keyword did not exist and all objects, both defined and undefined, were included in the list. Specify DEFINED ALL to get the behavior of DB2 9 and earlier.

**NO**

Specifies that only table spaces or index spaces that are currently undefined are included in the INCLUDED or EXCLUDED clause. Use EXCLUDE with DEFINED NO to remove table spaces and index spaces that are currently undefined and would not normally be processed by the utility. If you specify DEFINED NO, you cannot specify CLONED YES.
ALL
Specifies that table spaces or index spaces that are both undefined and defined are to be included in the INCLUDED or EXCLUDED clause.

Before DB2 10, the DEFINED keyword did not exist and all objects, both defined and undefined, were included in the list. Specify DEFINED ALL to get the behavior of DB2 9 and earlier.

RI
Specifies that all objects that are referentially related to the object expression (PRIMARY KEY <-> FOREIGN KEY) are to be included in the list. Db2 processes all referential relationships repeatedly until the entire referential set is developed. You cannot specify RI with PARTLEVEL(n).

Auxiliary indicator keywords: Use one of four auxiliary indicator keywords to direct LISTDEF processing to follow auxiliary relationships to include related LOB or XML objects in the list. The auxiliary relationship can be followed in either direction. Auxiliary objects include the auxiliary table spaces, auxiliary tables, indexes on auxiliary tables, and their containing index spaces.

Incomplete LOB or XML definitions cause seemingly related objects to not be found. The auxiliary relationship does not exist until you create the AUX TABLE with the STORES keyword.

No default auxiliary indicator keyword exists. If you do not specify BASE, LOB, XML, or ALL, Db2 does not follow the auxiliary relationships.

ALL
Specifies that BASE, LOB, and XML objects are to be included in the list. Auxiliary relationships are followed from all objects that result from the initial object lookup. BASE, LOB, and XML objects remain in the final enumerated list.

The behavior of the ALL keyword is altered by the presence or absence of the HISTORY or ARCHIVE keywords. When ALL is specified with HISTORY, the resulting list clause contains all related history objects. When ALL is specified with ARCHIVE, the resulting list clause contains all related archive objects (table spaces and index spaces that contain archive tables and their related indexes). When ALL is specified without HISTORY or ARCHIVE, the resulting list clause contains all related objects that are not history or archive objects.

BASE
Specifies that only base table spaces (non-LOB, non-XML) and index spaces are to be included in this element of the list. If the result of the initial search for the object is a base object, auxiliary relationships are not followed. If the result of the initial search for the object is a LOB or XML object, the auxiliary relationship is applied to the base table space or index space. Only those base objects become part of the resulting list.

The behavior of the BASE keyword is altered by the presence or absence of the HISTORY or ARCHIVE keywords. When BASE is specified with HISTORY, the resulting list clause contains only base history objects. When BASE is specified with ARCHIVE, the resulting list clause contains only base archive objects (base table spaces and index spaces that contain archive tables and their related indexes). When BASE is specified without HISTORY or ARCHIVE, the resulting list clause contains only base objects that are not history or archive objects.

LOB
Specifies that only LOB table spaces and related index spaces that contain
indexes on auxiliary tables are to be included in this element of the list. If the result of the initial search for the object is a LOB object, auxiliary relationships are not followed. If the result of the initial search for the object is a base object, the auxiliary relationship is applied to the LOB table space or index space. Only those LOB objects become part of the resulting list.

The behavior of the LOB keyword is altered by the presence or absence of the HISTORY or ARCHIVE keywords. When LOB is specified with HISTORY, the resulting list clause contains only LOB history objects (LOB table spaces and index spaces for history tables). When LOB is specified with ARCHIVE, the resulting list clause contains only LOB archive objects (LOB table spaces and index spaces for archive tables). When LOB is specified without HISTORY or ARCHIVE, the resulting list clause contains only LOB objects that are not history or archive objects.

XML

Specifies that only XML table spaces and related index spaces that contain indexes on auxiliary tables are to be included in this element of the list. If the result of the initial search for the object is an XML object, auxiliary relationships are not followed. If the result of the initial search for the object is a base object, the auxiliary relationship is applied to the XML table space or index space. Only those XML objects become part of the resulting list.

The behavior of the XML keyword is altered by the presence or absence of the HISTORY or ARCHIVE keywords. When XML is specified with HISTORY, the resulting list clause contains only XML history objects (XML table spaces and index spaces for history tables). When XML is specified with ARCHIVE, the resulting list clause contains only XML archive objects (XML table spaces and index spaces for archive tables). When XML is specified without HISTORY or ARCHIVE, the resulting list clause contains only XML objects that are not history or archive objects.

HISTORY

Specifies that only history (versioning) objects are to be included in the resulting list clause.

HISTORY is a filtering keyword that operates against the list clause contents after other keywords are applied. Use the keywords BASE, LOB, XML, or ALL with or without the HISTORY keyword to reference related objects. The order in which these keywords are specified has no meaning. Two INCLUDE or EXCLUDE clauses are required if both history and non-history objects are required.

ARCHIVE

Specifies that only archive objects are to be included in the resulting list clause.

ARCHIVE is a filtering keyword that operates against the list clause contents after other keywords are applied. Use the BASE, LOB, XML, or ALL keywords with or without the ARCHIVE keyword to reference related objects. The order in which these keywords are specified has no meaning. Two INCLUDE or EXCLUDE clauses are required if both archive and non-archive objects are required.

ARCHIVE cannot be specified with the HISTORY or CLONED YES keywords.

Related information:

[Archive-enabled tables and archive tables (Introduction to Db2 for z/OS)]

BASIC

Specifies that LISTDEF is to perform a final filtering of the INCLUDE or
EXCLUDE clause contents based on the basic format with 6-byte RBA or LRSN values. This operation is performed last, after LISTDEF processes all other keywords on the INCLUDE or EXCLUDE clause.

**YES**

Specifies that only table spaces and index spaces that are in basic format with 6-byte RBA or LRSN values are to be returned in the INCLUDE or EXCLUDE clause. This option is the same as EXTENDED NO.

**NO**

Specifies that only table spaces and index spaces that are not in basic format with 6-byte RBA or LRSN values format are to be returned in the INCLUDE or EXCLUDE clause. This option is the same as EXTENDED YES.

The BASIC keyword does not have a default value. If the RBA or LRSN format is not a factor, omit the BASIC and EXTENDED keywords.

If the clause also includes the PARTLEVEL keyword with either a non-zero operand or no operand, filtering is performed based on the RBA or LRSN format of the partition. If partitioned objects have partitions in different formats, PARTLEVEL must specify a non-zero operand or no operand to use BASIC or EXTENDED.

**EXTENDED**

Specifies that LISTDEF is to perform a final filtering of the INCLUDE or EXCLUDE clause contents based on the extended format with 10-byte RBA or LRSN values. This operation is performed last, after LISTDEF processes all other keywords on the INCLUDE or EXCLUDE clause.

**YES**

Specifies that only table spaces and index spaces that are in extended format with 10-byte RBA or LRSN values are to be returned in the INCLUDE or EXCLUDE clause. This option is the same as BASIC NO.

**NO**

Specifies that only table spaces and index spaces that are not in extended format with 10-byte RBA or LRSN values format are to be returned in the INCLUDE or EXCLUDE clause. This option is the same as BASIC YES.

The EXTENDED keyword does not have a default value. If the RBA or LRSN format is not a factor, omit the BASIC and EXTENDED keywords.

If the clause also includes the PARTLEVEL keyword with either a non-zero operand or no operand, filtering is performed based on the RBA or LRSN format of the partition. If partitioned objects have partitions in different formats, PARTLEVEL must specify a non-zero operand or no operand to use BASIC or EXTENDED.

---

**Concurrency and compatibility for LISTDEF**

The LISTDEF utility has certain concurrency and compatibility characteristics associated with it.

LISTDEF is a control statement that is used to set up an environment for another utility to follow. The LISTDEF list is stored until it is referenced by a specific utility. When referenced by an utility, the list expands. At that time, the concurrency and compatibility restrictions of that utility apply, with the additional restriction that the catalog tables that are necessary to expand the list must be available for read-only access.
List processing limitations

Although Db2 does not limit the number of objects that a list can contain, be aware that if your list is too large, the utility might fail with an error or abend in either Db2 or another program. These errors or abends can be caused by storage limitations, limitations of the operating system, or other restrictions imposed by either Db2 or non-Db2 programs. Whether such a failure occurs depends on many factors including, but not limited to the following items:

- The amount of available storage in both the utility batch and DBM1 address spaces
- The utility that is running.
- The type and number of other utilities that are running at the same time.
- The specific combination of keywords and operands of all the utilities that are running

Recommendation: If you receive a failure that you suspect is caused by running a utility on a list that is too large, divide your list into smaller lists and run the utility or utilities in separate job steps on the smaller lists until they run successfully.

Creating the LISTDEF control statement

The LISTDEF control statement defines a list of objects and assigns a name to the list.

Procedure

To create a LISTDEF control statement, include the following elements in the control statement:

- The name of the list.
- An INCLUDE clause, optionally followed by additional INCLUDE or EXCLUDE clauses to either include or exclude objects from the list.

Related concepts:

“How to include objects in a list”

Related reference:

“Syntax and options of the LISTDEF control statement” on page 215

How to include objects in a list

You can use the LISTDEF control statement options to specify which objects to include in the list. The INCLUDE clause adds objects to the list. The EXCLUDE clause removes objects from the list.

You must first specify an INCLUDE clause. You can then specify subsequent INCLUDE or EXCLUDE clauses in any order to add to or delete objects from the existing list.

Db2 constructs the list, one clause at a time, by adding objects to or removing objects from the list. If an EXCLUDE clause attempts to remove an object that is not yet in the list, Db2 ignores the EXCLUDE clause of that object and proceeds to the next INCLUDE or EXCLUDE clause. Be aware that a subsequent INCLUDE can return a previously excluded object to the list.
You must specify either INCLUDE or EXCLUDE. No default specification exists.

**Specifying objects to include or exclude**

Each INCLUDE or EXCLUDE clause identifies specific objects to add to or remove from the list.

You must include the following elements in each INCLUDE or EXCLUDE clause:

- The object that is to be used in the initial catalog lookup for each INCLUDE or EXCLUDE clause. The search for objects can begin with databases, table spaces, index spaces, tables, indexes, or other lists. You can explicitly specify the names of these objects or, with the exception of other lists, use a pattern matching expression. The resulting list contains only table spaces, only index spaces, or both.

- The type of objects that the list contains, either TABLESPACES or INDEXSPACES. You must explicitly specify the list type only when you specify a database as the initial object by using the keyword DATABASE. Otherwise, LISTDEF uses the default list type values shown in the following table. These values depend on the type of object that you specified for the INCLUDE or EXCLUDE clause.

<table>
<thead>
<tr>
<th>Specified object</th>
<th>Default list type value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLESPACE</td>
<td>TABLESPACES</td>
</tr>
<tr>
<td>TABLE</td>
<td>TABLESPACES</td>
</tr>
<tr>
<td>INDEXSPACE</td>
<td>INDEXSPACES</td>
</tr>
<tr>
<td>INDEX</td>
<td>INDEXSPACES</td>
</tr>
<tr>
<td>LIST</td>
<td>Existing type value of the list</td>
</tr>
</tbody>
</table>

For example, the following INCLUDE clause specifies that table space DBLT0301.TLLT031A is to be added to the LIST:

```
INCLUDE TABLESPACE DBLT0301.TLLT031A
```

In the preceding example, table space DBLT0301.TLLT031A is specified as the object that LISTDEF is to use for the initial catalog lookup. By default, the list type value for a TABLESPACE object is TABLESPACES. Therefore, the list includes only table space DBLT0301.TLLT031A.

The following example INCLUDE clause is similar to the preceding example, except that it includes the INDEXSPACES keyword:

```
INCLUDE INDEXSPACES TABLESPACE DBLT0301.TLLT031A
```

In this example, the clause specifies that all index spaces over all tables in table space DBLT0301.TLLT031A are to be added to the list.

Optionally, you can add related objects to the list by specifying keywords that indicate a relationship, such as referentially related objects or auxiliary related objects. Valid specifications include the following keywords:

- BASE (non-LOB and non-XML objects)
- LOB (LOB objects)
- XML (XML objects)
- ALL (BASE, LOB, and XML objects)
- TABLESPACES (related table spaces)
• INDEXSPACES (related index spaces)
• RI (related by referential constraints, including informational referential constraints)

The preceding keywords perform two functions: they determine which objects are related, and they then filter the contents of the list. The behavior of these keywords varies depending on the type of object that you specify. For example, if your initial object is a LOB object, the LOB keyword is ignored. If, however, the initial object is not a LOB object, the LOB keyword determines which LOB objects are related, and Db2 excludes non-LOB objects from the list.

Db2 processes each INCLUDE and EXCLUDE clause in the following order:
1. Perform the initial search for the object that is based on the specified pattern-matching expression, including PARTLEVEL specification, if specified. If an object is stopped, Db2 returns an error, even if the object is later excluded.
2. Add or remove related objects and filter the list elements based on the specified list type, either TABLESPACES or INDEXSPACES (COPY YES or COPY NO).
3. Add or remove related objects depending on the presence or absence of the RI, BASE, LOB, XML, and ALL keywords.

For example, to generate a list of all table spaces in the ACCOUNT database but exclude all LOB table spaces, you can specify the following LISTDEF statement:
LISTDEF ACCNT INCLUDE TABLESPACES DATABASE ACCOUNT BASE

In the preceding example, the name of the list is ACCNT. The TABLESPACES keyword indicates that the list is to include table spaces that are associated with the specified object. In this case, the table spaces to be included are those table spaces in database ACCOUNT. Finally, the BASE keyword limits the objects to only base table spaces.

If you want a list of only LOB index spaces in the ACCOUNT database, you can specify the following LISTDEF statement:
LISTDEF ACLOBIX INCLUDE INDEXSPACES DATABASE ACCOUNT LOB

In the preceding example, the INDEXSPACES and LOB keywords indicate that the INCLUDE clause is to add only LOB index spaces to the ACLOBIX list.

Restriction: Utilities do not support SYSUTILX-related objects inside a LISTDEF specification. You cannot specify the following objects in a LISTDEF:
• TABLESPACE DSNDB01.SYSUTILX
• TABLE SYSIBM.SYSUTILX
• TABLE SYSIBM.SYSUTIL
• INDEXSPACE DSNDB01.DSNLUX01
• INDEXSPACE DSNDB01.DSNLUX02
• INDEX SYSIBM.DSNLUX01
• INDEX SYSIBM.DSNLUX02

Using pattern matching expressions

You can use four special pattern-matching characters (%, *, _, ?) to define generic object names in a LISTDEF statement. These characters are similar to those characters that are used in the SQL LIKE predicate. Utilities that reference a list access the Db2 catalog at execution time and dynamically expand each generic
object name into an equivalent enumerated list. A utility processes this enumerated list either sequentially or in parallel, depending on the utility function and the parameters that you specify.

**Restrictions:** Db2 does not support all-inclusive lists (such as DATABASE * or TABLESPACE *.*).

Pattern-matching of Db2 catalog and directory objects (DSNDB06 and DSNDB01) is not supported. Catalog and directory objects must be included in a LISTDEF by their full table space or index space name. Even if catalog and directory objects match a LISTDEF pattern matching expression, they are not included in the list. To process those objects, you must use syntax from releases prior to Version 7.

Specify pattern-matching object names by using the pattern-matching characters that are shown in the following table. This table lists the pattern-matching character, the equivalent SQL symbol, and any additional information.

<table>
<thead>
<tr>
<th>LISTDEF pattern-matching character</th>
<th>Equivalent symbol used in SQL LIKE predicates</th>
<th>Usage notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percent sign (%)</td>
<td>Percent sign (%)</td>
<td>Performs the same function.</td>
</tr>
<tr>
<td>Question mark (?)</td>
<td>Underscore (_)</td>
<td>Use the question mark (?) instead of underscore (<em>) as a pattern-matching character in table and index names. The underscore character (</em>) in table and index names represents a single occurrence of itself.</td>
</tr>
<tr>
<td>Asterisk (*)</td>
<td>Percent sign (%)</td>
<td>Performs the same function.</td>
</tr>
<tr>
<td>Underscore (_)</td>
<td>Underscore (_)</td>
<td>Use the underscore (_) as an alternative to the question mark (?) for database, table space, and index space names.</td>
</tr>
</tbody>
</table>

**Including catalog and directory objects**

If you specify Db2 directory objects (DSNDB01) and Db2 catalog objects (DSNDB06) in object lists, you must specify the fully qualified table space or index space names for those objects. Pattern-matching is not supported for catalog or directory objects. Db2 issues error messages for any catalog or directory objects that are invalid for a utility.

Although Db2 catalog and directory objects can appear in LISTDEF lists, these objects might be invalid for a utility and result in an error message.

The following valid INCLUDE clauses contain catalog and directory objects:

- INCLUDE TABLESPACE DSNDB06.SYSDDF
- INCLUDE TABLESPACES TABLESPACE DSNDB06.SYSDDF
- INCLUDE INDEXSPACE DSNDB06.DSNDXX01
- INCLUDE INDEXSPACES INDEXSPACE DSNDB06.DSNDXX01

**Restriction:** If you specify a catalog or directory object in a LISTDEF control statement, you cannot specify the following keywords:

- DATABASE
- TABLE
- INDEX
- BASE
- LOB
- ALL
- Databases DSNDB01, DSNDB06, and DSNDB07
- Table or indexes with a creator id of SYSIBM

These keywords require Db2 to access the catalog, which can cause problems when you specify a catalog or directory object.

All LISTDEF lists automatically exclude work file databases, which consist of DSNDB07 objects and user-defined work file objects, because Db2 utilities do not process these objects.

### Previewing the contents of a list

You can preview the objects that are to be included in a list by using the PREVIEW function.

#### About this task

When you run a utility using the PREVIEW function, Db2 expands any LISTDEF control statements into the equivalent enumerated list, prints it to SYSPRINT, and stops execution.

#### Procedure

To preview the objects that are included in the list:

- Specify PREVIEW as a JCL parameter.
- Specify PREVIEW on the OPTIONS PREVIEW control statement.

#### Related concepts:

"Using the OPTIONS utility with LISTDEF” on page 233

#### Related reference:

Chapter 23, “OPTIONS,” on page 407

### Creating LISTDEF libraries

When Db2 encounters a reference to a list, Db2 first searches SYSIN. If Db2 does not find the definition of the referenced list, Db2 searches the specified LISTDEF library.

#### Procedure

To create a library of LISTDEF control statements:

Use a DD statement to name LISTDEF data sets.

For example, assume that data sets ADMF001.DB.LIST1 and ADMF001.DB.LIST2 each contain several LISTDEF statements. For any utility jobs that reference these LISTDEF statements, you can include the following DD statement in the JCL:

```plaintext
//LISTDSN DD DSN=ADMF001.DB.LIST1,DISP=SHR
// DD DSN=ADMF001.DB.LIST2,DISP=SHR
```

This DD statement defines a LISTDEF library. The statement gives a name (LISTDSN) to a group of data sets that contain LISTDEF statements, in this case ADMF001.DB.LIST1 and ADMF001.DB.LIST2. Defining such a library enables you to subsequently refer to the LISTDEF statements in that library by using the
OPTIONS LISTDEFDD control statement.
Any data sets that are identified as part of a LISTDEF library must contain only LISTDEF statements.
In the utility job that references those LISTDEF statements, include an OPTIONS statement before the utility statement. In the OPTIONS statement, specify the DD name of the LISTDEF library as LISTDEFDD ddname.
Db2 uses this LISTDEF library for any subsequent utility control statements, until either the end of input or until you specify another OPTIONS LISTDEFDD ddname. The default DD name for the LISTDEF definition library is SYSLISTD.

Referencing LISTDEF lists in other utility jobs

You can use a list of objects that was defined with a LISTDEF control statement as a target object for another utility.

Procedure

To reference LISTDEF lists in other utility jobs:

1. Specify LISTDEF control statements to define the lists of objects. You can specify these LISTDEF statements in one of the following places:
   - In the SYSIN DD statement before the utility control statement that references it
     
     Example:
     ```plaintext
     //SYSIN   DD *
     LISTDEF MYLIST INCLUDE TABLESPACES DATABASE PAYROLL
               INCLUDE INDEXSPACES DATABASE PAYROLL
     ```
   - In one or more LISTDEF library data sets
     
     Example:
     ```plaintext
     //****************************************************
     // Create an input data set.
     //****************************************************
     //LOAD1    EXEC PGM=IEBGENER
     //SYSPRINT DD DUMMY
     //SYSIN    DD DUMMY
     //SYSUT2   DD DSN=JULUT103.TCASE.DATA2,
               DISP=(NEW,CATLG,CATLG),
               UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND),
               DCB=(RECFM=FB,LRECL=80,BLksz=2400)
     //SYSUT1   DD *
     LISTDEF NAME1 INCLUDE TABLESPACE DBLT0301.TLLT031A
               INCLUDE TABLESPACE DBLT0301.TSLT031B
     ```

     Any LISTDEF statement that is defined within the SYSIN DD statement overrides another LISTDEF definition of the same name in a LISTDEF library data set.

2. If you want to reference a list that is defined in a LISTDEF library data set, use the OPTIONS utility. In the SYSIN DD statement before the utility control statement that references the list, specify OPTIONS LISTDEFDD with the name of the LISTDEF library.

     Example:
     ```plaintext
     //****************************************************
     // QUIESCE LISTDEF DD LILSTDEF data sets
     //****************************************************
     //STEP1   EXEC DSNUPROC,UID='JULUT103.QUIESC2',
     ```
3. In the utility control statement, specify the LIST keyword and the list name.
For example, you can use the QUIESCE utility to quiesce all objects in a list by specifying the following control statement:

```
QUIESCE LIST list-name
```

Some utilities such as COPY and RECOVER, can process a LIST without a specified object type. Object types are determined from the list contents. Other utilities, such as REPORT, RUNSTATS, and REORG INDEX, must know the object type that is to be processed before processing can begin. These utilities require that you specify an object type in addition to the LIST keyword (for example: REPORT RECOVERY TABLESPACE LIST, RUNSTATS INDEX LIST, and REORG INDEX LIST). See the syntax diagrams for an individual utility for details.

## Results

In general, utilities process the objects in the list in the order in which they are specified. However, some utilities alter the list order for optimal processing. The following table shows the utilities that support the LIST keyword and how each utility processes the list.

<table>
<thead>
<tr>
<th>Utility</th>
<th>Order of list processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHECK INDEX</td>
<td>Items are grouped by related table space. All index spaces that are related to a particular table space are processed at one time, regardless of list order.</td>
</tr>
<tr>
<td>COPY</td>
<td>Items are processed in the specified order on a single call to COPY. The PARALLEL keyword is supported for a list, but if used, the order of processing is determined by Db2.</td>
</tr>
<tr>
<td>COPYTOCOPY</td>
<td>Items are processed in the specified order on a single call to COPYTOCOPY.</td>
</tr>
<tr>
<td>MERGECOPY</td>
<td>Items are processed in the specified order.</td>
</tr>
<tr>
<td>MODIFY RECOVERY</td>
<td>Items are processed in the specified order.</td>
</tr>
<tr>
<td>MODIFY STATISTICS</td>
<td>Items are processed in the specified order.</td>
</tr>
<tr>
<td>QUIESCE</td>
<td>All items are processed in the specified order on a single call to QUIESCE.</td>
</tr>
<tr>
<td>REBUILD</td>
<td>Items are grouped by related table space. All index spaces that are related to a particular table space are processed at one time, regardless of list order.</td>
</tr>
<tr>
<td>RECOVER</td>
<td>Items are processed in the specified order on a single call to RECOVER.</td>
</tr>
<tr>
<td>REORG</td>
<td>Items are processed in the specified order with one exception. Items at the partition level are grouped by table space when the first partition of a particular table space is encountered. Those partitions are processed on a single call to REORG.</td>
</tr>
<tr>
<td>REPORT</td>
<td>Items are processed in the specified order.</td>
</tr>
<tr>
<td>RUNSTATS INDEX</td>
<td>Items are grouped by related table space. All index spaces that are related to a particular table space are processed at one time, regardless of list order.</td>
</tr>
</tbody>
</table>
Utility | Order of list processing
--- | ---
RUNSTATS TABLESPACE | Items are processed in the specified order.
UNLOAD | Items at the partition level are grouped by table space. All specified partitions of a particular table space are processed at one time, regardless of list order.

**Related tasks:**
- “Creating LISTDEF libraries” on page 230

**Related reference:**
- “Syntax and options of the LISTDEF control statement” on page 215
- “Sample LISTDEF control statements” on page 234
- “Syntax and options of the OPTIONS control statement” on page 407

---

**Using the TEMPLATE utility with LISTDEF**

Together, the LISTDEF and TEMPLATE utilities enable faster development of utility job streams, and require fewer modifications when the underlying list of database objects change.

Many utilities require output data sets. In those cases, you should use the TEMPLATE control statement to specify the naming convention and, optionally, the allocation parameters for each type of output data set. Templates, like lists, can be reused if the naming convention is robust enough to prevent duplicate data set names from being allocated.

In some cases you can use traditional JCL DD statements with LISTDEF lists, but this method is usually not practical unless you are processing small lists one object at a time.

**Related reference:**
- Chapter 34, “TEMPLATE,” on page 793

---

**Using the OPTIONS utility with LISTDEF**

You can use the OPTIONS utility with LISTDEF.

Use the following three functions of the OPTIONS utility in conjunction with the LISTDEF utility when needed:

**OPTIONS PREVIEW**
- Enables you to preview the list contents before actual processing.

**OPTIONS ITEMERROR**
- Enables you to alter the handling of errors that might occur during list processing.

**OPTIONS LISTDEFDD**
- Enables you to identify a LISTDEF library. The default value is LISTDEFDD.

**Related tasks:**
- “Creating LISTDEF libraries” on page 230

---

**Termination or restart of LISTDEF**

You can terminate and restart the LISTDEF utility.
You can terminate a LISTDEF utility job by using the TERM UTILITY command if you submitted the job or have SYSOPR, SYSCTRL, or SYSADM authority.

You can restart a LISTDEF utility job, but it starts from the beginning again. Use caution when changing LISTDEF lists prior to a restart. When Db2 restarts list processing, it uses a saved copy of the list. Modifying the LISTDEF list that is referred to by the stopped utility has no effect. Only control statements that follow the stopped utility are affected.

**Related tasks:**

"Restarting an online utility" on page 44

---

### Sample LISTDEF control statements

Use the sample control statements as models for developing your own LISTDEF control statements.

#### Example 1: Defining a list of objects

The following control statement defines a list that includes the following objects:

- Table space DBLT0301.TLLT031A
- Index space DBLT0301.IXIT031A
- Table space DBLT0301.IPLT031C
- Table space that contains ADMF001.TBLT032A_1

The name of the list is NAME1. This list can be referenced by any subsequent utility statements.

```
LISTDEF NAME1 INCLUDE TABLESPACE DBLT0301.TLLT031A
   INCLUDE INDEXSPACE DBLT0301.IXIT031A
   INCLUDE TABLESPACE DBLT0301.IPLT031C
   INCLUDE TABLESPACES TABLE ADMF001.TBLT032A_1
```

#### Example 2: Defining a list of all objects in a database

The following control statement defines a list (EXAMPLE2) that includes all table spaces and all index spaces in the PAYROLL database.

```
LISTDEF EXAMPLE2 INCLUDE TABLESPACES DATABASE PAYROLL
   INCLUDE INDEXSPACES DATABASE PAYROLL
```

#### Example 3: Defining a list by using pattern-matching characters

The following control statement defines a list (PAYROLL) that includes the following objects:

- All table spaces in the PAYROLL database, except for any table spaces whose names begin with TEMP.
- All index spaces in the PAYROLL database that end with IX, except for those index spaces that begin with TMPIX.

The subsequent COPY utility control statement processes this list.

```
LISTDEF PAYROLL INCLUDE TABLESPACE PAYROLL.*
   EXCLUDE TABLESPACE PAYROLL.TEMP*
   INCLUDE INDEXSPACE PAYROLL.*IX
   EXCLUDE INDEXSPACE PAYROLL.TMPIX*
COPY LIST PAYROLL ...
```
Example 4: Defining a list of partitions and nonpartitioned table spaces

The following LISTDEF statement defines a list that includes one entry for each partition of the qualifying partitioned table spaces and one entry for each qualifying nonpartitioned table space. The list is named EXAMPLE4. The table spaces must satisfy the PAY*.* name pattern.

```
LISTDEF EXAMPLE4 INCLUDE TABLESPACE PAY*.* PARTLEVEL
```

Assume that three table spaces qualify. Of these table spaces, two are partitioned table spaces (PAY2.DEPTA and PAY2.DEPTF) that each have three partitions and one is a nonpartitioned table space (PAY1.COMP). In this case, the EXAMPLE4 list includes the following items:

- PAY2.DEPTA partition 1
- PAY2.DEPTA partition 2
- PAY2.DEPTA partition 3
- PAY2.DEPTF partition 1
- PAY2.DEPTF partition 2
- PAY2.DEPTF partition 3
- PAY1.COMP

If you specified PARTLEVEL(2) instead of PARTLEVEL, the EXAMPLE4 list includes the following items:

- PAY2.DEPTA partition 2
- PAY2.DEPTF partition 2
- PAY1.COMP

If you specified PARTLEVEL(0) instead of PARTLEVEL, the EXAMPLE4 list includes only PAY1.COMP.

Example 5: Defining a list of COPY YES indexes

The following control statement defines a list (EXAMPLE5) that includes related index spaces from the referenced list (EXAMPLE4) that were defined or altered to COPY YES.

```
LISTDEF EXAMPLE5 INCLUDE LIST EXAMPLE4 INDEXSPACES COPY YES
```

Example 6: Defining a list that includes all table space partitions except for one

The following control statement defines a list (EXAMPLE6) that includes all partitions of table space X, except for partition 12. The INCLUDE clause adds an entry for each partition, and the EXCLUDE clause removes the entry for partition 12.

```
LISTDEF EXAMPLE6 INCLUDE TABLESPACE X PARTLEVEL
   EXCLUDE TABLESPACE X PARTLEVEL(12)
```

If the PARTLEVEL keyword is not specified in both clauses, as in the following two sample statements, the INCLUDE and EXCLUDE items do not intersect. For example, in the following statement, table space X is included in the list in its entirety, not at the partition level. Therefore, partition 12 cannot be excluded.

```
LISTDEF EXAMPLE6 INCLUDE TABLESPACE X
   EXCLUDE TABLESPACE X PARTLEVEL(12)
```
In the following sample statement, the list includes only partition 12 of table space X, so table space X in its entirety cannot be excluded.

LISTDEF EXAMPLE6 INCLUDE TABLESPACE X PARTLEVEL(12)
EXCLUDE TABLESPACE X

Example 7: Defining a LISTDEF library and referencing a list in a QUIESCE job

In this example, the first two LISTDEF control statements define the NAME1 and NAME2 lists. The NAME1 list is stored in a sequential data set (JULTU103.TCASE.DATA2). The NAME2 list is stored in a member of a partitioned data set (JULTU103.TCASE.DATA3(MEM1)). These output data sets are identified by the SYSUT2 DD statements (in the JCL for the CREATE1 and CREATE2 jobs).

The LISTLIB DD statement (in the JCL for the QUIESCE job) defines a LISTDEF library. When you define a LISTDEF library, you give a name to a group of data sets that contain LISTDEF statements. In this case, the library is to include the following data sets:

- The sequential data set JULTU103.TCASE.DATA2 (which includes the NAME1 list)
- The MEM1 member of the partitioned data set JULTU103.TCASE.DATA3 (which includes the NAME2 list).

When you define such a library, you can later reference a group of LISTDEF statements with a single reference.

The OPTIONS utility statement in this example specifies that the library that is identified by the LISTLIB DD statement is to be used as the default LISTDEF definition library. This declaration means that for any referenced lists, Db2 is to first search SYSIN for the list definition. If Db2 does not find the list definition in SYSIN, it is to search any data sets that are included in the LISTLIB LISTDEF library.

The last LISTDEF statement defines the NAME3 list. This list includes all objects in the NAME1 and NAME2 lists, except for three table spaces (TSLT032B, TSLT031B, TSLT032C). Because the NAME1 and NAME2 lists are not included in SYSIN, Db2 searches the default LISTDEF library (LISTLIB) to find them.

Finally, the QUIESCE utility control statement specifies this list of objects (NAME3) for which Db2 is to establish a quiesce point.

```sql
/*CREATE1 JOB 'USER=NAME',CLASS=A,...
**-----------------------------------------------
** Create an input data set.
**-----------------------------------------------
*/LOAD1 EXEC PGM=IEBGENER
/*SYSPRINT DD DUMMY
/SYSIN DD DUMMY
/SYSUT2 DD DSN=JULTU103.TCASE.DATA2,
// DISP=(NEW,CATLG,CATLG),
// UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=2400)
/SYSUT1 DD *
LISTDEF NAME1 INCLUDE TABLESPACE DBLT0301.TLLT031A
INCLUDE TABLESPACE DBLT0301.TSLT031B
*/
/*CREATE2 JOB 'USER=NAME',CLASS=A,...
**-----------------------------------------------
** Create an input data set.
*/
```
Example 8: Defining a list that includes related objects

The following LISTDEF control statement defines a list (EXAMPLE8) that includes table space DBLT0101.TPLT011C and all objects that are referentially related to it. Only base table spaces are included in the list. The subsequent RECOVER utility control statement specifies that all objects in the EXAMPLE8 list are to be recovered.

Example 9: Defining a list of cloned data

The following control statement indicates that the INCLUDE expression is to return only the names of the following objects:

- Clone tables
- Table spaces that contain clone tables
- Indexes on clone tables
• Index spaces that contain indexes on clone tables

LISTDEF REORG_TBSP INCLUDE TABLESPACE DB42240*.T*  
      CLONED YES  
      EXCLUDE TABLESPACE DB42240*.TL4224L*  
      EXCLUDE TABLESPACE DB42240*.TL4224B*  
      EXCLUDE TABLESPACE DB42240*.TL4224C*  
      EXCLUDE TABLESPACE DB42240*.TL4224D*  
      EXCLUDE TABLESPACE DB42240*.TL4224E*  
      EXCLUDE TABLESPACE DB42240*.TL4224F*  
      EXCLUDE TABLESPACE DB422401.TSHR5702

Example 10: Defining a list that includes archive objects

The following LISTDEF statement defines a list with the name LISTALL that includes all related table spaces, including related archive table spaces.

LISTDEF LISTALL  
      INCLUDE TABLESPACES TABLESPACE DB516A01.TU516A01 RI ALL  
      INCLUDE TABLESPACES TABLESPACE DB516A01.TU516A01 RI ALL ARCHIVE

The first INCLUDE clause specifies that all base, LOB, and XML table spaces that are referentially related to the table space DB516A01.TU516A01 are to be included in the list. The second INCLUDE clause specifies that all archive table spaces that are related to table space DB516A01.TU516A01 are to be included in the list.
Chapter 19. LOAD

Use the LOAD online utility to load one or more tables of a table space. The LOAD utility loads records into the tables and builds or extends any indexes that are defined on them.

If the table space already contains data, you can choose whether you want to add the new data to the existing data or replace the existing data.

The loaded data is processed by any edit or validation routine that is associated with the table, and any field procedure that is associated with any column of the table. The LOAD utility ignores and does not enforce informational referential constraints.

To avoid the cost of running the RUNSTATS utility afterward, you can also specify the STATISTICS option collect inline statistics when you run the LOAD utility.

You can use the LOAD utility in conjunction with z/OS DFSMS data set encryption with the REPLACE option to encrypt or decrypt table spaces or indexes that use Db2-managed data sets. The LOAD utility accepts encrypted input data sets.

Output

LOAD DATA generates one or more of the following forms of output:

- A loaded table space or partition.
- A discard file of rejected records.
- A summary report of errors that were encountered during processing; this report is generated only if you specify ENFORCE CONSTRAINTS or if the LOAD utility involves unique indexes.

The output can be encrypted if a key label is defined for the output data set.

Authorization required

To execute this utility, you must use a privilege set that includes one of the following authorizations:

- Ownership of the table
- LOAD privilege for the database
- STATS privilege for the database is required if STATISTICS keyword is specified
- DBADM or DBCTRL authority for the database. If the database is implicitly created, these privileges must be on the implicitly created database or on DSNDB04.
- DATAACCESS authority
- SYSCTRL or SYSADM authority

The LOAD utility operates on a table space level, so you must have authority for all tables in the table space when you perform LOAD.

To run LOAD STATISTICS, the privilege set must include STATS authority on the database. To run LOAD STATISTICS REPORT YES, the privilege set must also include the SELECT privilege on the tables required.
If either the FLASHCOPY YES or FLASHCOPY CONSISTENT option is specified, the user ID that invokes the LOAD utility must have the authority to execute the DFSMSdss COPY command.

If you use RACF access control with multilevel security and the LOAD utility is to process a table space that contains a table that has multilevel security with row-level granularity, you must be identified to RACF and have an accessible valid security label. You must also meet the following authorization requirements:

- To replace an entire table space with LOAD REPLACE, you must have the write-down privilege unless write-down rules are not in effect.
- You must have the write-down privilege to specify values for the security label columns, unless write-down rules are not in effect. If these rules are in effect and you do not have write-down privilege, Db2 assigns your security label as the value for the security label column for the rows that you are loading.

To run LOAD on an encrypted data set, you must be authorized to use the key label for that data set.

**Restrictions on running LOAD**

- LOAD with REPLACE cannot be run on a table space during the period after RECOVER is run to a point in time before materialization of pending definition changes and before REORG is run to complete the point-in-time recovery process.

**Execution phases of LOAD**

The LOAD utility operates in the following phases:

**UTILINIT**
Performs initialization.

**RELOAD**

Loads record types and writes temporary file records for indexes and foreign keys. RELOAD makes one pass through the sequential input data set. Check constraints are checked for each row. Internal commits provide commit points at which to restart in case operation should halt in this phase.

RELOAD creates inline copies if you specified the COPYDDN or RECOVERYDDN keywords.

A subtask is started at the beginning of the RELOAD phase to sort the keys. The sort subtask initializes and waits for the main RELOAD phase to pass its keys to SORT. RELOAD loads the data, extracts the keys, and passes them in memory for sorting. At the end of the RELOAD phase, the last key is passed to SORT, and record sorting completes.

Note that load partition parallelism starts subtasks. PREFORMAT for table spaces occurs at the end of the RELOAD phase.

**SORT**

Sorts temporary file records before creating indexes or validating referential constraints, if indexes or foreign keys exist. The SORT phase is skipped if all the following conditions apply for the data that is processed during the RELOAD phase:

- Each table has no more than one key.
- All keys are the same type (index key only, indexed foreign key, or foreign key only).
• The data that is being loaded or reloaded is in key order (if a key exists). If the key is an index key only and the index is a data-partitioned secondary index, the data is considered to be in order if the data is grouped by partition and ordered within partition by key value. If the key in question is an indexed foreign key and the index is a data-partitioned secondary index, the data is never considered to be in order.
• The data that is being loaded or reloaded is grouped by table, and each input record is loaded into one table only.

SORT passes the sorted keys in memory to the BUILD phase, which builds the indexes.

BUILD
Creates indexes from temporary file records for all indexes that are defined on the loaded tables. Build also detects duplicate keys. PREFORMAT for indexes occurs at the end of the BUILD phase.

SORTBLD
Performs all activities that normally occur in both the SORT and BUILD phases, if you specify a parallel index build.

INDEXVAL
Corrects unique index violations or index evaluation errors from the information in SYSERR, if any exist.

ENFORCE
Checks referential constraints, except informational referential constraints, and corrects violations. Information about violations of referential constraints is stored in SYSERR.

DISCARD
Copies records that cause errors from the input data set to the discard data set.

REPORT
Generates a summary report, if you specified ENFORCE CONSTRAINT or if load index validation is performed. The report is sent to SYSPRINT.

SWITCH
Switches access to shadow copy of table space or partition. This phase occurs if you specify LOAD REPLACE SHRLEVEL REFERENCE.

LOGAPPLY
Identifies the most recent checkpoint for each member. All objects that are being copied are updated to the same log point to prepare for backout processing.

If FLASHCOPY CONSISTENT is specified, Db2 applies the updates to the FlashCopy image copy to ensure that copy includes all activity up to the point of consistency.

LOGCSR
Calls log apply do the current status rebuild (CSR).

If FLASHCOPY CONSISTENT is specified, the utility reads the logs during this phase. The utility uses the logs to identify the uncommitted work that needs to be backed out of the image copy.

LOGUNDO
Backs out uncommitted work from the image copy.
If FLASHCOPY CONSISTENT is specified, the utility backs out uncommitted work from the FlashCopy image copy to make the image copy consistent.

**UTILTERM**

Performs cleanup.

Related concepts:

- [Multilevel security (Managing Security)](#)

Related tasks:

- [Maintaining Db2 database statistics (Db2 Performance)](#)
- [Reducing the cost of collecting statistics (Db2 Performance)](#)

### Syntax and options of the LOAD control statement

The LOAD utility control statement, with its multiple options, defines the function that the utility job performs.

You can create a control statement with the ISPF/PDF edit function. After creating it, save it in a sequential or partitioned data set. When you create the JCL for running the job, use the SYSIN DD statement to specify the name of the data set that contains the utility control statement.
Syntax diagram

Notes:
1 The default is 0 if the input is on tape, a cursor, a PDS member or for SYSREC DD *.
   For sequential data sets on disk, LOAD computes the default based on the input data set size.

resume-spec:
Notes:
1 The RESUME or REPLACE value in the INTO TABLE clause overrides the default value for LOAD RESUME.

workddn-spec:

COPYDDN:

RECOVERYDDN:

flashcopy-spec:

statistics-spec:
stat-table-spec

```
TABLE(ALL)
  SAMPLE integer
  TABLE(tableName)
```

table-stats-spec:

```
COLUMN ALL
  SAMPLE integer
  COLUMN columnName

, 

COLGROUP columnName
colgroup-stats-spec
```

colgroup-stats-spec:

```
FREQVAL COUNT 10 MOST COUNT integer
  MOST
  BOTH
  LEAST

HISTOGRAM NUMQUANTILES 100
  NUMQUANTILES integer
```

stat-index-spec
**correlation-stats-spec:**

<table>
<thead>
<tr>
<th>(1)</th>
<th>KEYCARD</th>
<th>FREQVAL</th>
<th>NUMCOLS</th>
<th>COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FREQVAL</th>
<th>NUMCOLS</th>
<th>COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>integer</td>
<td>integer</td>
</tr>
</tbody>
</table>

**HISTOGRAM**

<table>
<thead>
<tr>
<th>NUMCOLS</th>
<th>NUMQUANTILES</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>100</td>
</tr>
</tbody>
</table>

**Notes:**

1. The KEYCARD option is deprecated. The KEYCARD functionality is now built into the default execution of the inline statistics for indexes and cannot be disabled.

**format-spec:**

<table>
<thead>
<tr>
<th>FORMAT</th>
<th>UNLOAD</th>
<th>SQL/DS</th>
<th>INTERNAL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DELIMITED</th>
<th>CHARDEL</th>
<th>DECPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>'comma'</td>
<td>'single quote'</td>
<td>'.'</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SPANNED</th>
<th>YES</th>
<th>NO</th>
</tr>
</thead>
</table>

**ignore-spec:**
decfloat-spec:

\[
\begin{array}{c}
\text{DECFLOAT_ROUNDMODE} \\
\text{ROUND_CEILING} \\
\text{ROUND_DOWN} \\
\text{ROUND_FLOOR} \\
\text{ROUND_HALF_DOWN} \\
\text{ROUND_HALF_EVEN} \\
\text{ROUND_HALF_UP} \\
\text{ROUND_UP}
\end{array}
\]

override-spec:

\[
\begin{array}{c}
\text{OVERRIDE} \\
\text{SYSTEMPERIOD} \\
\text{IDENTITY} \\
\text{TRANSID} \\
\text{NONDETERMINISTIC}
\end{array}
\]

Notes:

1. **Alternate syntax:** The following syntax is still accepted: PERIODOVERRIDE instead of OVERRIDE(SYSTEMPERIOD), TRANSIDOVERRIDE instead of OVERRIDE(TRANSID), IDENTITYOVERRIDE instead of OVERRIDE(IDENTITY). However, the keywords PERIODOVERRIDE, TRANSIDOVERRIDE, and IDENTITYOVERRIDE are deprecated.
Notes:
1  The default for DRAIN_WAIT is the value of the IRLMRWT subsystem parameter.
2  The default for RETRY is the value of the UTIMOUT subsystem parameter.
3  The default for RETRAY_DELAY is the smaller of the following two values: \( \min(DRAIN\_WAIT\ \text{value} \times \text{RETRY\_value}, DRAIN\_WAIT\ \text{value} \times 10) \)

labeled-duration-expression

INTO-TABLE-spec:
For the syntax diagram and the option descriptions of the into-table specification, see “INTO-TABLE-spec” on page 277.

Option descriptions

DATA
  Specifies that data is to be loaded. This keyword is optional and is used for clarity only.

INDDN ddname
  Specifies the input data set.
  
  \textit{ddname} is the name of a DD statement from the job step or the name of a template in the utility input stream. The default value is \texttt{SYSREC}. 
The record format for an input data set must be fixed-length or variable-length. The data set must be a sequential data set that is readable by the basic sequential access method (BSAM).

If the input file is an HFS or zFS file, use a template with the PATH option.

**Related information:**

“Syntax and options of the TEMPLATE control statement” on page 793

**INCURSOR cursor-name**

Specifies that the input data set is to be the result table of a SELECT statement, as defined by a cursor. You must declare the cursor before it is used by LOAD. Use the EXEC SQL utility to define the cursor.

`cursor-name` is the name of the cursor. This name cannot be longer than eight characters. The specified cursor can be used with the Db2® family cross-loader function, which enables you to load data from any DRDA-compliant remote server.

**Restrictions:**

- You cannot load data into the same table on which you defined the cursor.
- You cannot load data into the parent table in an RI relationship by using a cursor that is defined on a dependent table.
- You cannot specify field specifications or use discard processing with the INCURSOR option.
- You cannot specify INCURSOR with the following options
  - SHRLEVEL CHANGE
  - NOSUBS
  - FORMAT UNLOAD
  - FORMAT SQL/DS
  - FORMAT INTERNAL
  - CONTINUEIF
  - WHEN
  - SPANNED YES

**Recommendation:** Do not specify a cursor on a table within the same table space as the table that you are loading. If you cannot avoid this situation, disable SQL parallelism by specifying the following EXEC SQL statement:

```sql
EXEC SQL
SET CURRENT DEGREE = '1' ;
ENDEXEC
```

Disabling parallelism in this case helps avoid contention between claims and drains.

**Related information:**

Chapter 17, “EXEC SQL,” on page 211

“Loading data by using the cross-loader function” on page 333

**PREFORMAT**

Specifies that the remaining pages are preformatted up to the high-allocated RBA in the table space and index spaces that are associated with the table that is specified in `table-name`. The preformatting occurs after the data has been loaded and the indexes are built.

PREFORMAT can operate on an entire table space and its index spaces, or on a partition of a partitioned table space and on the corresponding partitions of
partitioned indexes, if any exist. Specifying LOAD PREFORMAT (rather than
PART integer PREFORMAT) tells LOAD to serialize at the table space level,
which can inhibit concurrent processing of separate partitions. If you want to
serialize at the partition level, specify PART integer PREFORMAT.

The PREFORMAT keyword also applies to LOB table spaces and auxiliary
indexes that are associated with the base table or partitions that LOAD
serialized. XML objects are not preformatted.

COPYDICTIONARY integer
Allows the LOAD utility to copy an existing compression dictionary from a
partition to other partitions of a partitioned table space. LOAD copies the
current compression dictionary from the partition whose partition number is
integer, and uses that compression dictionary to compress the input data for
partitions that are being replaced. The default value of integer is 1.
COPYDICTIONARY provides a method for copying a compression dictionary
to an empty partition. The partition that is being copied must have a valid
compression dictionary.

COPYDICTIONARY causes LOAD to copy the compression dictionary only to
partitions that have the COMPRESS YES attribute.

Use of the COPYDICTIONARY keyword has these restrictions:

- COPYDICTIONARY can be used only when the target of the LOAD
  statement is a partitioned (non-UTS) or partition-by-range table space.
- PART integer REPLACE must also be specified in the LOAD statement.
- RESUME YES cannot be specified with COPYDICTIONARY.
- KEEPDICTIONARY cannot also be specified in the LOAD statement.

PRESORTED
Specifies whether the input data set has already been sorted in clustering key
order. If the input data set is in clustering key order, the LOAD utility can
execute the RELOAD and BUILD phases in parallel, and can skip the sorting
of the clustering index.

NO Specifies that the input data set has not already been sorted. The LOAD
utility must sort the clustering index.

YES Specifies that the input data set has already been sorted. The LOAD utility
does not sort the clustering index, and executes the RELOAD and BUILD
phases in parallel.

The following requirements must be satisfied when PRESORTED YES is
specified:

- All data sets that are needed for parallel index build need to be
  available.
- For partitioned table spaces with a clustering partitioned index, the
  presorted order of the data rows must be:
  1. By partition number
  2. By key ordering of clustering index within each partition
- For partitioned table spaces with a clustering nonpartitioned index, or
  nonpartitioned table space with a single table, the presorted order of the
data rows must be by key ordering of the clustering index.
- For simple and segmented table spaces:
  - The presorted order of the data rows must be by key ordering of the
    clustering index within the table.
The LOAD statement can contain only one INTO TABLE clause.

Restrictions:

- Under the following conditions, LOAD issues a warning message, and continues with processing as if PRESORTED NO were specified:
  - When SHRLEVEL CHANGE is also specified
  - When partition parallelism is used
  - When the target tables have no indexes
  - When SORTKEYS NO is specified
- Only LOAD with REPLACE and with PRESORTED YES can be restarted in the RELOAD phase. If LOAD with RESUME and PRESORTED YES is restarted in the RELOAD phase, utility processing abnormally terminates, and LOAD issues an error message.
- If PRESORTED YES is specified, and LOAD determines that the input data set is not sorted in clustering key order, LOAD tolerates the keys that are not in order. However, for the clustering index, inline statistics are not collected and real-time statistics are invalidated. LOAD issues a warning message.

**PARALLEL or PARALLEL num-subtasks**

Specifies the maximum number of subtasks that LOAD can process in parallel. By using parallel subtasks, the utility can potentially reduce the elapsed time of the load operation.

If you are loading from a single input data set, PARALLEL enables additional data parallelism.

`num-subtasks` must be an integer between 0 and 32767, inclusive.

**Recommendation:** Specify PARALLEL(0) or PARALLEL.

If you specify the PARALLEL keyword without a value, the default value is PARALLEL(0).

The total possible number of subtasks for a particular LOAD job depends on the following factors:

- the number of data partitions to be loaded
- the number of indexes on the table to be loaded
- whether inline statistics are gathered
- whether the LOAD statement specifies a single input data set for the entire job or one input data set per partition.

LOAD calculates the optimal number of these subtasks to process in parallel based on memory constraints and the number of available processors.

LOAD uses the value of PARALLEL as follows:

- **If you specify PARALLEL(0) or PARALLEL:** LOAD uses the value that it calculated for the optimal number of parallel subtasks.
- **If you specify PARALLEL(1):** The utility loads the data with the minimal amount of parallel subtasks that are needed for the load operation.
- **If you specify a value for PARALLEL other than 0 or 1:** If the value is less than the calculated optimal number of parallel subtasks, LOAD uses the specified value or the minimum number of required parallel subtasks. If the value is greater than the calculated optimal number of parallel subtasks, LOAD limits the number of parallel subtasks to the optimal number.
PARALLEL overrides the value of the PARAMDEG_UTIL subsystem parameter.

**Recommendation:** If you specify PARALLEL and SHRLEVEL CHANGE, set the LOCKSIZE attribute of the table space to ROW to minimize contention on the parallel subtasks.

**Restrictions:** PARALLEL is not valid in the following situations:
- A single input data set is specified and the LOAD statement includes any of the following options:
  - SPANNED YES
  - INCURSOR
  - PRESORTED
  - FORMAT INTERNAL
  - COLGROUP
- The table space to be loaded is a partition-by-growth table space and the LOAD statement includes the SHRLEVEL NONE option.
- The table to be loaded has XML columns and is in a simple or segmented table space, and the LOAD statement includes the SHRLEVEL CHANGE option.
- The table to be loaded has LOB or XML columns, and the LOAD statement includes the SHRLEVEL NONE option.

**Related information:**
- MAX UTILS PARALLELISM field (PARAMDEG_UTIL subsystem parameter) (Db2 Installation and Migration)
- Specifying the size of locks for a table space (Db2 Performance)
- “Improving LOAD performance” on page 336

**ROWFORMAT**
Specifies the output row format in the affected table space or partition. This keyword overrides the existing RRF subsystem parameter setting. This keyword has no effect on LOB, catalog, directory, XML, or universal (UTS) table spaces participating in a CLONE relationship.

- **BRF**
  Specifies that the table space or partition being reorganized or replaced will be converted to or remain in basic row format.

- **RRF**
  Specifies that the table space or partition being reorganized or replaced will be converted to or remain in reorder row format.

**RBALRSN_CONVERSION**
Specifies the RBA or LRSN format of the target object after the completion of the LOAD utility. If the keyword is not specified, the conversion specified in the UTILITY_OBJECT_CONVERSION subsystem parameter is accepted. If you specify RBALRSN_CONVERSION, you must also specify the REPLACE keyword.

- **NONE**
  Specifies that no conversion is performed.
The utility fails if RBALRSN_CONVERSION NONE is specified on a table space that is in basic 6-byte format and the
UTILITY_OBJECT_CONVERSION subsystem parameter is set to NOBASIC.

**BASIC**

Specifies that if an object is found in extended 10-byte format, it is converted to 6-byte basic format.

The utility fails if RBALRSN_CONVERSION BASIC is specified and the
UTILITY_OBJECT_CONVERSION subsystem parameter is set to NOBASIC.

**EXTENDED**

Specifies that if an object is found in basic 6-byte format, it is converted to 10-byte extended format.

If RBALRSN_CONVERSION is not specified, and subsystem parameter
UTILITY_OBJECT_CONVERSION is set to BASIC or EXTENDED,
RBALRSN_CONVERSION defaults to the UTILITY_OBJECT_CONVERSION value. If RBALRSN_CONVERSION is not specified, and subsystem parameter
UTILITY_OBJECT_CONVERSION is set to NOBASIC,
RBALRSN_CONVERSION defaults to EXTENDED.

LOAD REPLACE of a base table space, when converting page format to extended, does not convert versioned XML table spaces that are associated with that base table space.

**RESUME**

Indicates whether records are to be loaded into an empty or non-empty table space. For nonsegmented table spaces, space is not reused for rows that have been marked as deleted or for rows of dropped tables.

**Important:** Specifying LOAD RESUME (rather than PART integer RESUME) tells LOAD to serialize on the entire table space, which can inhibit concurrent processing of separate partitions. If you want to process other partitions concurrently, specify PART integer RESUME.

**NO**

Loads records into an empty table space. If the table space is not empty, and you have not used REPLACE, a message is issued and the utility job step terminates with a job step condition code of 8.

For nonsegmented table spaces that contain deleted rows or rows of dropped tables, using the REPLACE keyword provides increased efficiency.

The default value is NO, unless you override it with PART integer RESUME YES.

**YES**

Loads records into a non-empty table space. If the table space is empty, a warning message is issued, but the table space is loaded. Loading begins at the current end of data in the table space. Space is not reused for rows that are marked as deleted or for rows of dropped tables.

RESUME YES is not valid with SHRLEVEL REFERENCE.

LOAD RESUME YES SHRLEVEL CHANGE activates the before triggers and after triggers for each row that is loaded.

If LOAD RESUME YES SHRLEVEL NONE is specified with COPYDDN or RECOVERYDDN, an inline image copy will be created during LOAD processing. The inline image copy will be a full copy for the table space. If
individual partitions are specified using INTO TABLE PART, the image

copy and SYSCOPY records will only include those partitions.

**BACKOUT**

Specifies whether to delete all rows loaded by the current LOAD operation if
any record would leave the object unavailable. YES is the default value if the
BACKOUT keyword is specified alone. However, no BACKOUT processing
occurs unless a non-zero DISCARDS value is specified and the number of
discarded records exceeds the DISCARDS value.

BACKOUT or BACKOUT YES requires RESUME YES to be specified at the
table space level, or in the resume-spec of the INTO TABLE PART specification.
If RESUME YES is specified in the INTO TABLE PART specification, it must be
specified on each INTO TABLE PART specification.

BACKOUT or BACKOUT YES is supported only with SHRLEVEL NONE.
BACKOUT or BACKOUT YES is not supported with INCURSOR.

**YES**

Specifies that all rows loaded by the current LOAD operation are
deleted if any input record would leave the object unavailable. The
table space is available at the completion of the LOAD. YES is the
default when BACKOUT is specified.

**NO**

Specifies that updates made by LOAD are not be rolled back if an error
record is found. This behavior is the default when the BACKOUT
keyword is not specified.

**SHRLEVEL**

Specifies the extent to which applications can concurrently access the table
space or partition during the LOAD utility job. The following parameter values
are listed in order of increasing extent of allowed concurrent access.

**NONE**

Specifies that applications have no concurrent access to the table space or
partition.

**REFERENCE**

Specifies that applications can concurrently read from the table space or
partition into which LOAD is loading data, except for the duration of the
SWITCH phase.

LOAD REPLACE SHRLEVEL REFERENCE specifies that data is reloaded
into a shadow copy of the target objects, and LOAD switches the future
access of an application from the original copy to the shadow copy in the
SWITCH phase. This option is not supported on table spaces with clone
relationship, or on target table defined with LOB or XML column.

**CHANGE**

Specifies that applications can concurrently read from and write to the
table space or partition into which LOAD is loading data for the majority
of the LOAD duration.

A LOAD SHRLEVEL CHANGE job functions like a mass INSERT. Whereas
a regular LOAD job drains the entire table space, LOAD SHRLEVEL
CHANGE functions like an INSERT statement and uses claims when it
accesses an object.

If you specify SHRLEVEL CHANGE, you cannot specify any of the
following parameters:

- INCURSOR
- RESUME NO
If you are loading individual partitions and specify SHRLEVEL CHANGE, RESUME YES must also be specified. You can either specify RESUME YES on the individual PART clauses or inherit it from the main LOAD statement.

LOAD RESUME YES SHRLEVEL CHANGE does not perform the SORT, BUILD, SORTBLD, INDEXVAL, or ENFORCE phases, and the compatibility and concurrency considerations differ.

Normally, a LOAD RESUME YES job loads the records at the end of the already existing records. However, for a LOAD RESUME YES job with the SHRLEVEL CHANGE option, the utility tries to insert the new records in available free space as close to the clustering order as possible. This LOAD job does not create any additional free pages. If you insert many records, these records are likely to be stored out of clustering order. In this case, you should run the REORG TABLESPACE utility after the LOAD utility loads the records.

**Recommendation:** If the LOAD utility loads many records, run RUNSTATS SHRLEVEL CHANGE UPDATE SPACE and then a conditional REORG.

When an identity column exists in the table that is being loaded, performance can be improved by specifying the CACHE attribute for the identity column.

Lock escalation is disabled on XML table spaces for LOAD RESUME YES SHRLEVEL CHANGE.

Log records that Db2 creates during LOAD RESUME YES SHRLEVEL CHANGE can be used by Db2 DataPropagator, if the tables that are being loaded are defined with DATA CAPTURE CHANGES.

LOAD jobs with the SHRLEVEL CHANGE option do not insert any records into SYSIBM.SYSCOPY.

Before and after row triggers are activated only for SHRLEVEL CHANGE. Statement triggers for each row are also activated for LOAD RESUME YES SHRLEVEL CHANGE.

**REPLACE**

Indicates whether the table space and all its indexes need to be reset to empty before records are loaded. With this option, the newly loaded rows replace all existing rows of all tables in the table space, not just those of the table that you are loading. For Db2 STOGROUP-defined data sets, the data set is deleted and redefined with this option, unless you also specified the REUSE option. You
must have LOAD authority for all tables in the table space where you perform LOAD REPLACE. If you attempt a LOAD REPLACE without this authority, you get an error message.

You cannot use REPLACE with the PART integer REPLACE option of INTO TABLE; you must either replace an entire table space by using the REPLACE option or replace a single partition by using the PART integer REPLACE option of INTO TABLE.

Specifying LOAD REPLACE (rather than PART integer REPLACE) tells LOAD to serialize at the table space level. If you want to serialize at the partition level, specify PART integer REPLACE. See the information about specifying REPLACE at the partition level under the keyword descriptions for INTO TABLE.

Restrictions:

- LOAD REPLACE is not allowed on a table that is defined with data versioning.
- LOAD REPLACE is not allowed on a table space after RECOVER was run on that table space to a point in time before pending definition changes were materialized. Before running LOAD REPLACE, you need to run REORG on the entire table space to complete the point-in-time recovery process.
- LOAD REPLACE is not allowed on an archive-enabled table. (LOAD REPLACE is allowed on the table space that contains the archive table.)

"Replacing data with LOAD" on page 315

COPYDDN (ddname1, ddname2)

Specifies the DD statements for the primary (ddname1) and backup (ddname2) copy data sets for the image copy.

ddname is the DD name.

The default value is SYSCOPY for the primary copy. No default exists for the backup copy.

The COPYDDN keyword can be specified with REPLACE or with RESUME YES SHRLEVEL NONE.

If COPYDDN is specified with REPLACE, a full image copy data set (SHRLEVEL REFERENCE) is created for the table or partitions that are specified when LOAD executes. The table space or partition for which an image copy is produced is not placed in COPY-pending status.

If COPYDDN is specified with RESUME YES SHRLEVEL NONE, an inline image copy will be created during LOAD processing. The inline image copy will be a full copy for the table space. If individual partitions are specified using INTO TABLE PART, the image copy and SYSCOPY records will only include those partitions. If COPYDDN is specified at the table space level, it cannot be specified at the INTO TABLE PART level.

Image copies that are taken during LOAD REPLACE are not recommended for use with RECOVER TOCOPY because these image copies might contain unique index violations, referential constraint violations, or index evaluation errors.

If you specify COPYDDN when loading a table with XML data, an inline copy is taken only of the base table space, not the XML table space.
If you specify COPYDDN when loading a table with LOB columns, Db2 does not create a copy of any index, LOB table space, or XML table space. You must perform these tasks separately.

The COPYDDN keyword specifies either a DD name or a TEMPLATE name specification from a previous TEMPLATE control statement. If utility processing detects that the specified name is both a DD name in the current job step and a TEMPLATE name, the utility uses the DD name.

**RECOVERYDDN  ddname3, ddname4**

Specifies the DD statements for the primary (ddname3) and backup (ddname4) copy data sets for the image copy at the recovery site.

`ddname` is the DD name.

You cannot have duplicate image copy data sets. The same rules apply for RECOVERYDDN and COPYDDN.

If RECOVERYDDN is specified at the table space level, it cannot be specified at the INTO TABLE PART level.

The RECOVERYDDN keyword specifies either a DD name or a TEMPLATE name specification from a previous TEMPLATE control statement. If utility processing detects that the specified name is both a DD name in the current job step and a TEMPLATE name, the utility uses the DD name.

**FLASHCOPY**

Specifies whether FlashCopy technology is used to create a copy of the object. Valid values are YES, NO, or CONSISTENT. When FlashCopy is used, a separate data set is created for each partition or piece of the object.

Specify YES or CONSISTENT only if the Db2 data sets are on FlashCopy Version 2 disk volumes.

The FlashCopy specifications on the utility control statement override any specifications for FlashCopy that are defined by using the Db2 subsystem parameters. If the FlashCopy subsystem parameters specify the use of FlashCopy as the default behavior of this utility, the FLASHCOPY option can be omitted from the utility control statement.

**Important:** If the input data set is less than one cylinder, FlashCopy technology might not be used for copying the objects regardless of the FLASHCOPY settings. The copy is performed by IDCAMS if FlashCopy is not used.

**NO**  Specifies that no FlashCopy is made. NO is the default value for FLASHCOPY.

**YES**  Specifies that FlashCopy technology is used to copy the object.

**Important:** Under the following circumstances, the COPY utility might not use FlashCopy even though YES is specified:

- FlashCopy Version 2 disk volumes are not available
- The source tracks are already the target of a FlashCopy operation
- The target tracks are the source of a FlashCopy operation
- The maximum number of relationships for the copy is exceeded

In the event that FlashCopy is not used, the LOAD utility uses traditional I/O methods to copy the object, which can result in longer than expected execution time.
CONSISTENT

When SHRLEVEL CHANGE is specified, specifies that FlashCopy technology is used to copy the object and that any uncommitted work included in the copy is backed out of the copy to make the copy consistent. If SHRLEVEL NONE is specified, the image copy is already consistent and you do not need to specify CONSISTENT.

A consistent FlashCopy image copy can by used for recovery without also requiring a sequential format image copy.

Specifying FLASHCOPY CONSISTENT requires additional time and system resources during utility processing, because the utility must read the logs and apply the changes to the image copy. Similarly, recovering from a consistent FlashCopy image copy also requires additional time and system resources to read the logs and reapply work that was previously backed out.

Restriction: CONSISTENT cannot be specified when copying objects that have been defined with the NOT LOGGED attribute. If CONSISTENT is specified for an object that is defined with the NOT LOGGED attribute, the utility does not make a copy of the object and issues message DSNU076I with return code 8.

Related information:
- “FlashCopy image copies” on page 157
- “Subsystem parameters for refining DFSMSdss COPY operation with utilities” on page 40

FCCOPYDDN

Specifies the template to be used to create the FlashCopy image copy data set names. If a value is not specified for FCCOPYDDN on the LOAD control statement when FlashCopy is used, the value specified on the FCCOPYDDN subsystem parameter determines the template to be used.

(template-name)

The data set names for the FlashCopy image copy are allocated according to the template specification. For table space or index space level FlashCopy image copies, because a data set is allocated for each partition or piece, ensure that the data set naming convention in the template specification is unique enough. Use the &DSNUM variable, which resolves to a partition number or piece number at execution time.

STATISTICS

Specifies the gathering of statistics for a table space, index, or both; the statistics are stored in the Db2 catalog.

If you specify the STATISTICS keyword with no other statistics-spec or correlation-stats-spec options, the utility gathers only table space statistics. Statistics are collected on a base table space, but not on a LOB table space or XML table space.

Restrictions:
- If you specify STATISTICS for encrypted data, Db2 might not provide useful statistics on this data.
- You cannot specify STATISTICS if the named table is a table clone.
All tables identified by the STATISTICS TABLE keyword must belong to the table space of the specified table or tables in the INTO TABLE option of the LOAD utility control statement.

**TABLE**

Specifies the table for which column information is to be gathered.

Do not specify STATISTICS TABLE *table-name* with the LIST keyword. Instead, specify STATISTICS TABLE (ALL).

(ALL) Specifies that information is to be gathered for all columns of all tables in the table space.

*table-name* Specifies the tables for which column information is to be gathered. If you omit the qualifier, the user identifier for the utility job is used. Enclose the table name in quotation marks if the name contains a blank.

If you specify more than one table, you must repeat the TABLE option. Multiple TABLE options must be specified entirely before or after any INDEX keyword that may also be specified. For example, the INDEX keyword may not be specified between any two TABLE keywords.

**SAMPLE integer**

Indicates the percentage of rows to be sampled when collecting statistics on non-leading-indexed columns of an index or non-indexed columns. You can specify any value from 1 through 100.

The default value is 25. The SAMPLE option is not allowed for LOB table spaces.

**COLUMN**

Specifies columns for which column information is to be gathered.

You can specify this option only if you specify a particular table for which statistics are to be gathered (TABLE *(table-name)*). If you specify particular tables and do not specify the COLUMN option, the default, COLUMN(ALL), is used. If you do not specify a particular table when using the TABLE option, you cannot specify the COLUMN option; however, COLUMN(ALL) is assumed.

(ALL) Specifies that statistics are to be gathered for all columns in the table.

*(column-name, ...)* Specifies the columns for which statistics are to be gathered.

You can specify a list of column names; the maximum is 10. If you specify more than one column, separate each name with a comma.

**INDEX**

Specifies indexes for which information is to be gathered. Column information is gathered for the first column of the index. All the indexes must be associated with the same table space, which must be the table space that is specified in the TABLESPACE option.

Do not specify STATISTICS INDEX *index-name* with the LIST keyword. Instead, specify STATISTICS INDEX (ALL).

(ALL) Specifies that the column information is to be gathered for all indexes that are defined on tables that are contained in the table space.
Specifies the indexes for which information is to be gathered. Enclose the index name in quotation marks if the name contains a blank.

**COLGROUP** (column-name, ...)
Indicates that the specified set of columns are treated as a group. This option enables inline statistics to collect a cardinality value on the specified column group. Inline statistics ignores COLGROUP when processing XML table spaces and indexes.

When you specify the COLGROUP keyword, inline statistics collects correlation statistics for the specified column group. If you want inline statistics to also collect distribution statistics, specify the FREQVAL option with COLGROUP.

(column-name, ...) specifies the names of the columns that are part of the column group.

When you define a column group on a single column, you can potentially improve RUNSTATS performance by specifying the STATCLGMEMSRT option or changing the value of the STATCLGSRT subsystem parameter. You can use these options to avoid column group sorts by external sort programs.

To specify more than one column group, repeat the COLGROUP option.

**Restriction:** The length of the COLGROUP value cannot exceed the maximum length of the COLVALUE column in the SYSIBM.SYSCOLDIST catalog table.

**Related information:**
STATISTICS COLGROUP DATA SORT STG LIMIT field (STATCLGSRT subsystem parameter) (Db2 Installation and Migration)

**FREQVAL**
Indicates, when specified with the COLGROUP option, that frequency statistics are also to be gathered for the specified group of columns. (COLGROUP indicates that cardinality statistics are gathered.) One group of statistics is gathered for each column. You must specify COUNT integer with COLGROUP FREQVAL. The utility ignores FREQVAL MOST/LEAST/BOTH when processing XML table spaces and indexes.

**COUNT integer**
Indicates the number of frequently occurring values to be collected from the specified column group. For example, COUNT 20 means that Db2 collects 20 frequently occurring values from the column group. When the COUNT keyword is not specified, the default is COUNT 10. Specifying a value of 1000 or more can increase the prepare time for some SQL statements.

**MOST**
Indicates that the utility is to collect the most frequently occurring values for the specified set of columns when COLGROUP is specified.

**BOTH**
Indicates that the utility is to collect the most and the least frequently occurring values for the specified set of columns when COLGROUP is specified.

**LEAST**
Indicates that the utility is to collect the least frequently occurring values for the specified set of columns when COLGROUP is specified.
**HISTOGRAM**
Indicates, when specified with the COLGROUP option, that histogram statistics are to be gathered for the specified group of columns. Inline statistics ignore HISTOGRAM when processing XML table spaces and indexes.

Histogram statistics that you collect through inline statistics are not the same as histogram statistics that you collect through RUNSTATS. Histogram statistics that you collect with inline statistics are only rough estimates. To obtain more exact statistics, use RUNSTATS.

**NUMQUANTILES**  integer
Indicates how many quantiles that the utility collects. The integer value must be greater than or equal to one. The number of quantiles that you specify must never exceed the total number of distinct values in the column or the column group. The maximum number of quantiles is 100.

When the NUMQUANTILES keyword is omitted, NUMQUANTILES takes a default value of 100. Based on the number of records in the table, the number of quantiles is readjusted down to an optimal number.

**KEYCARD**
The KEYCARD option is deprecated in the utility control statement and no longer needs to be specified to collect cardinality statistics on the values in the key columns of an index.

When the STATISTICS and INDEX options are specified, the utility always collects all of the distinct values in all of the 1 to n key column combinations in an index. n is the number of columns in the index. With the deprecation of KEYCARD, this functionality cannot be disabled.

The utility tolerates the specification of the KEYCARD option. The utility does not issue any messages if the control statement includes or excludes the KEYCARD option when STATISTICS and INDEX are specified.

**FREQVAL**
Controls the collection of frequent-value statistics. If you specify FREQVAL, it must be followed by the NUMCOLS keyword.

**NUMCOLS**
Indicates the number of key columns that are to be concatenated together when collecting frequent values from the specified index. Specifying ‘3’ means that frequent values are to be collected on the concatenation of the first three key columns. The default value is 1, which means that Db2 collects frequent values on the first key column of the index.

**COUNT**
Indicates the number of frequent values that are to be collected. Specifying ‘15’ means that Db2 collects 15 frequent values from the specified key columns. The default value is 10.

**HISTOGRAM**
Indicates that histogram statistics are requested for the specified index.

**NUMCOLS**
The number of key columns that are to be concatenated when collecting histogram statistics from the specified index.

**NUMQUANTILES**
The integer values that follows NUMQUANTILES indicates the number quantiles are requested. The integer value must be greater than or equal to 1.
Histogram statistics can be collected only on keys with the same order if the specified key columns for histogram statistics are of mixed order, a DSNU633I warning message is issued.

**Related information:**

**REPORT**
Specifies whether a set of messages is to be generated to report the collected statistics.

**NO** Indicates that the set of messages is not to be sent as output to SYSPRINT.

**YES** Indicates that the set of messages is to be sent as output to SYSPRINT. The generated messages are dependent on the combination of keywords (such as TABLESPACE, INDEX, TABLE, and COLUMN) that are specified with the RUNSTATS utility. However, these messages are not dependent on the specification of the UPDATE option. REPORT YES always generates a report of SPACE and ACCESSPATH statistics.

**UPDATE**
Indicates whether the collected statistics are to be inserted into the catalog tables. UPDATE also allows you to select statistics that are used for access path selection or statistics that are used by database administrators.

**ALL** Indicates that all collected statistics are to be updated in the catalog.

**ACCESSPATH**
Indicates that only the catalog table columns that provide statistics that are used for access path selection are to be updated.

**SPACE**
Indicates that only the catalog table columns that provide statistics to help database administrators assess the status of a particular table space or index are to be updated.

**NONE** Indicates that no catalog tables are to be updated with the collected statistics. This option is valid only when REPORT YES is specified.

**STATCLGMEMSRT integer**
Specifies the amount of memory that the utility can use for sorting records when collecting statistics on a single column that is defined with the COLGROUP option. Use STATCLGMEMSRT to avoid column group sorts by an external sort program, which can negatively affect the performance of statistics collection.

*integer* specifies the number of megabytes of memory space that the utility can use for an in-memory sort. Valid values are 0 to 4096. If the amount of space that is needed for the sort exceeds the *integer* value, the utility invokes a sort program. If you specify 0, the utility automatically invokes a sort program.

The amount of space that is needed for the column group sort depends on the following factors:

- The number of column groups for which the utility is collecting statistics
- The length of the single-column column group
- The number of distinct values in the column (cardinality)
The value of STATCLGMEMSRT overrides the value of the STATCLGSRT subsystem parameter.

Related information:
STATISTICS COLGROUP DATA SORT STG LIMIT field (STATCLGSRT subsystem parameter) (Db2 Installation and Migration)

HISTORY
Specifies that all catalog table inserts or updates to the catalog history tables are to be recorded.

The default value is whatever value is specified in the STATISTICS HISTORY field on panel DSNTIP6.

ALL
Indicates that all collected statistics are to be updated in the catalog history tables.

ACCESSPATH
Indicates that only the catalog history table columns that provide statistics that are used for access path selection are to be updated.

SPACE
Indicates that only space-related catalog statistics are to be updated in catalog history tables.

NONE
Indicates that no catalog history tables are to be updated with the collected statistics.

FORCEROLLUP
Specifies whether aggregation or rollup of statistics is to take place when RUNSTATS is executed even if statistics have not been gathered on some partitions; for example, partitions have not had any data loaded. Aggregate statistics are used by the optimizer to select the best access path.

YES
Indicates that forced aggregation or rollup processing is to be done, even though some partitions might not contain data.

NO
Indicates that aggregation or rollup is to be done only if data is available for all partitions.

If data is not available for all partitions, DSNU623I message is issued if the installation value for STATISTICS ROLLUP on panel DSNTIP6 is set to NO.

KEEPDICTIONARY
Prevents the LOAD utility from building a new compression dictionary. LOAD retains the current compression dictionary and uses it for compressing the input data. This option eliminates the cost that is associated with building a new dictionary.

The KEEPDICTIONARY keyword is ignored for XML table spaces. If you specify REPLACE, any existing dictionary for the XML table space or partition is deleted. If you do not specify REPLACE, any existing dictionary for the XML table space or partition is saved.

Db2 ignores the KEEPDICTIONARY option during execution of a REORG or LOAD REPLACE that changes the table space from basic row format to reordered row format.
This keyword is valid only if the table space that is being loaded has the COMPRESS YES attribute.

If the table space or partition is empty, Db2 performs one of these actions:

- Db2 builds a dictionary if a compression dictionary does not exist, but only if the table space is not a simple table space.
- Db2 keeps the dictionary if a compression dictionary exists.

If RESUME NO and REPLACE are specified when the table space or partition is not empty, Db2 performs the same actions as it does when the table space or partition is empty.

If the table space or partition is not empty and RESUME YES is specified, Db2 performs one of these actions:

- Db2 does not build a dictionary if a compression dictionary does not exist.
- Db2 keeps the dictionary if a compression dictionary exists.

**Note:** You must use KEEPDICTIONARY to ensure that the compression dictionary is maintained.

**Related information:**

"Compressing data by using the LOAD utility" on page 330

**REUSE**

Specifies (when used with REPLACE) that LOAD is to logically reset and reuse Db2-managed data sets without deleting and redefining them. If you do not specify REUSE, Db2 deletes and redefines Db2-managed data sets to reset them.

REUSE must be accompanied by REPLACE to do the logical reset for all data sets. However, if you specify REUSE for the table space and REPLACE only at the partition level, only the replaced partitions are logically reset.

If a data set has multiple extents, the extents are not released if you specify the REUSE parameter.

**LOG**

Indicates whether logging occurs during the RELOAD phase of the load process for LOAD SHRLEVEL NONE and LOAD RESUME YES SHRLEVEL CHANGE execution. For LOAD RESUME YES SHRLEVEL REFERENCE, and LOAD REPLACE SHRLEVEL REFERENCE, LOG NO always applies.

**YES**

Specifies normal logging during the load process. All records that are loaded are logged. If the table space has the NOT LOGGED attribute, Db2 does the LOAD with no logging.

**NO**

Specifies no logging of data during the load process. If the table space has the LOGGED attribute, the NO option sets the COPY-pending restriction against the table space or partition that the loaded table resides in. No table or partition in the table space can be updated by SQL until the restriction is removed. For ways to remove the restriction, see "Resetting COPY-pending status" on page 356.

If you load a single partition of a partitioned table space and the table space has a secondary index, some logging might occur during the build phase as Db2 logs any changes to the index structure. This logging allows recoverability of the secondary index in case an abend occurs, and it also allows concurrency.
Db2 treats table spaces that were created as NOT LOGGED as if you specified LOG NO. If you specify LOG NO without specifying COPYDDN, the base table space is placed in COPY-pending status. If XML columns are nullable and not loaded, only the base table space is placed in COPY-pending status.

A LOB table space affects logging while Db2 loads a LOB column regardless of whether the LOB table space was defined with LOG YES or LOG NO.

**NOCOPYPEND**

Specifies that LOAD is not to set the table space in the COPY-pending status, even though LOG NO was specified. A NOCOPYPEND specification does not turn on or change any informational COPY-pending (ICOPY) status for indexes. A NOCOPYPEND specification will not turn off any COPY-pending status that was set prior to the LOAD. Normal completion of a LOAD LOG NO NOCOPYPEND job returns a 0 code if no other errors or warnings exist.

Db2 ignores a NOCOPYPEND specification if you also specified COPYDDN to make a local-site inline image copy during the LOAD. If the table space has the NOT LOGGED attribute, NOCOPYPEND is ignored.

**Attention:** Specify the NOCOPYPEND option only if the data in the table space can be easily re-created by another LOAD job if the data is lost. If you do not take an image copy following the LOAD, you cannot recover the table space by using the RECOVER utility, and you might lose data.

**WORKDDN (ddname1,ddname2)**

Specifies the DD statements for the temporary work file for sort input and sort output. Temporary work files for sort input and output are required if the LOAD involves tables with indexes.

*ddname1* is the DD name for the temporary work file for sort input. The default value is **SYSUT1**.

*ddname2* is the DD name for the temporary work file for sort output. The default value is **SORTOUT**.

The WORKDDN keyword specifies either a DD name or a TEMPLATE name specification from a previous TEMPLATE control statement. If utility processing detects that the specified name is both a DD name in the current job step and a TEMPLATE name, the utility uses the DD name. For more information about TEMPLATE specifications, see Chapter 34, “TEMPLATE,” on page 793.

**SORTKEYS**

Specifies that index keys are to be sorted in parallel during the SORTBLD phase to improve performance.

*integer*

Specifies an integer to provide an estimate of the number of index keys that are to be sorted. Integer must be a positive integer between 0 and 562 949 953 421 311. The default value is 0 if any of the following conditions are true:

- The target table has no index and SHRLEVEL is NONE.
- The target table has one index.
• The input is on tape, a cursor, a PDS member, or for SYSREC DD *.

**NO** Indicates that the SORTKEYS default is to be turned off.

For sequential data sets on disk, LOAD computes the default value for SORTKEYS based on the input data set size.

If the NUMRECS keyword is specified at the table level in the same LOAD statement, you cannot specify an integer value on the SORTKEYS keyword. To turn off parallel sorts, you can specify SORTKEYS NO when the NUMRECS keyword is specified. The NUMRECS keyword is specified at the table level to improve the estimation of work file data set sizes when loading data into multi-table table spaces with uneven distribution of rows between the tables.

**Related information:**

"Improving LOAD performance” on page 336

**FORMAT**

Identifies the format of the input record. If you use FORMAT UNLOAD, FORMAT INTERNAL, or FORMAT SQL/DS, it uniquely determines the format of the input, and no field specifications are allowed in an INTO TABLE option.

If you omit FORMAT, the format of the input data is determined by the rules for field specifications. If you specify FORMAT DELIMITED, the format of the input data is determined by the rules that are described in Appendix G, “Delimited file format,” on page 1125.

**UNLOAD**

Specifies that the input record format is compatible with the Db2 unload format. (The Db2 unload format is the result of REORG with the UNLOAD ONLY option.)

Input records that were unloaded by the REORG utility are loaded into the tables from which they were unloaded, if an INTO TABLE option specifies each table. Do not add columns or change column definitions of tables between the time you run REORG UNLOAD ONLY and LOAD FORMAT UNLOAD.

Any WHEN clause on the LOAD FORMAT UNLOAD statement is ignored; Db2 reloads the records into the same tables from which they were unloaded. Not allowing a WHEN clause with the FORMAT UNLOAD clause ensures that the input records are loaded into the proper tables. Input records that cannot be loaded are discarded.

If the DCB RECFM parameter is specified on the DD statement for the input data set, and the data set format has not been modified since the REORG UNLOAD (ONLY) operation, the record format must be variable (RECFM=V).

**SQL/DS**

Specifies that the input record format is compatible with the SQL/DS unload format. The data type of a column in the table that is to be loaded must be the same as the data type of the corresponding column in the SQL/DS table.

If the SQL/DS input contains rows for more than one table, the WHEN clause of the INTO TABLE option indicates which input records are to be loaded into which Db2 table.

LOAD cannot load SQL/DS strings that are longer than the Db2 limit.
SQL/DS data that has been unloaded to disk under DB2 Server for VSE & VM resides in a simulated z/OS-type data set with a record format of VBS. Consider this format when transferring the data to another system that is to be loaded into a Db2 table (for example, the DB2 Server for VSE & VM. FILEDEF must define it as a z/OS-type data set). Processing the data set as a standard CMS file puts the SQL/DS record type field at the wrong offset within the records; LOAD is unable to recognize them as valid SQL/DS input.

**INTERNAL**

Specifies that the input record format is in Db2 internal format. Db2 internal format is the format that is produced by running UNLOAD with the FORMAT INTERNAL option.

**Attention:** FORMAT INTERNAL is a performance option that bypasses all data validity checks. Incorrect use might result in serious problems, including data overlays and Db2 abnormal terminations. Those problems might occur during LOAD processing or when the target table is accessed after it is loaded.

When FORMAT INTERNAL is specified:

- LOAD ignores any field specifications in the LOAD control statement.
- LOAD does no data conversion.

**Restrictions:**

- The definition of the table from which the input data was unloaded must match the definition of the table into which the data is loaded exactly. If a column of the table into which data is loaded has a field procedure, the corresponding column in the table from which the data was unloaded must also have that field procedure.
- The input data must be in decompressed format.
- LOAD can load only one table at a time when FORMAT INTERNAL is specified.
- LOAD does not populate LOB or XML columns when FORMAT INTERNAL is specified. LOAD puts the base table space in advisory CHECK-pending status.
- FORMAT INTERNAL cannot be specified with any of the following options:
  - ASCII
  - CCSID
  - CONTINUEIF
  - DECFLOAT_ROUNDMODE
  - EBCDIC
  - FLOAT
  - OVERRIDE(IDENTITY)
  - IGNOREFIELDS
  - INCURSOR
  - NOSUBS
  - SHRLEVEL CHANGE
  - UNICODE
  - WHEN
DELIMITED

Specifies that the input data file is in a delimited format. When data is in a delimited format, all fields in the input data set are character strings or external numeric values. In addition, each column in a delimited file is separated from the next column by a column delimiter character.

For each of the delimiter types that you can specify, you must ensure that the delimiter character is specified in the code page of the source data. The delimiter character can be specified as either a character or hexadecimal constant. For example, to specify '#' as the delimiter, you can specify either COLDEL ' # ' or COLDEL X'23'. If the utility statement is coded in a character type that is different from the input file, such as a utility statement that is coded in EBCDIC and input data that is in Unicode, you should specify the delimiter character in the utility statement as a hexadecimal constant, or the result can be unpredictable.

You cannot specify the same character for more than one type of delimiter (COLDEL, CHARDEL, and DECPT). If the input data includes external date, time, and timestamp formats, the separators cannot use the same character that is used for a delimiter. For more information about delimiter restrictions, see "Loading delimited files" on page 325.

Unicode input data for FORMAT DELIMITED must be UTF-8, CCSID 1208.

If you specify the FORMAT DELIMITED option, you cannot use any of the following options:

- CONTINUEIF
- INCURSOR
- Multiple INTO TABLE statements
- WHEN

Also, LOAD ignores any specified POSITION statements within the LOAD utility control statement.

For more information about using delimited output and delimiter restrictions, see "Loading delimited files" on page 325. For more information about delimited files see Appendix G, "Delimited file format," on page 1125.

COLDEL coldel

Specifies the column delimiter that is used in the input file. The default value is a comma (,). For most ASCII and UTF-8 data, this value is X'2C', and for most EBCDIC data, this value is a X'6B'.

CHARDEL chardel

Specifies the character string delimiter that is used in the input file. The default value is a double quotation mark ("). For most ASCII and UTF-8 data, this value is X'22', and for most EBCDIC data, this value is X'7F'.

To delimit character strings that contain the character string delimiter, repeat the character string delimiter where it is used in the character string. LOAD interprets any pair of character delimiters that are found between the enclosing character delimiters as a single character. For example, the phrase "what a ""nice warm"" day" is interpreted as what a "nice warm" day. The LOAD utility recognizes these character delimiter pairs for only CHAR, VARCHAR, and CLOB fields.

Character string delimiters are required only when the string contains the CHARDEL character. However, you can put the character string delimiters around other character strings. Data that has been unloaded
in delimited format by the UNLOAD utility includes character string delimiters around all character strings.

**DECPT**

Specifies the decimal point character that is used in the input file. The default value is a period (.). For most ASCII and UTF-8 data, this value is X'2E', and for most EBCDIC data, this value is X'4B'.

**Note:** If you use an application defaults load module (either DSNHDECP, which is the default, or a user-specified application defaults load module), ensure that the specified decimal value is the same as the decimal value that is used in the input data. You must specify the decimal value to match the decimal value that is used in the input data.

**SPANNAED**

Indicates whether records are to be loaded from a VBS data set in spanned record format.

**YES**

Indicates that the LOAD utility is to load data from spanned records.

The input data set must be in spanned record format and all LOB and XML data must be at the end of the record.

You must provide a field specification list with all LOB and XML fields at the end of the record. For LOB and XML columns, specify POSITION(*).

If you specify FORMAT SPANNED YES, do not reference LOB or XML data in the field-selection-criterion of a WHEN clause.

You cannot specify the INCURSOR option with SPANNED YES.

If you specify FORMAT SPANNED YES, the LOAD utility does not use parallel processing.

**NO**

Indicates that the LOAD utility is not to load data in spanned record format.

**Related information:**

| Unloading data in spanned record format” on page 866 |

**FLOAT**

Specifies that LOAD is to expect the designated format for floating point numbers.

**(S390)**

Specifies that LOAD is to expect that floating point numbers are provided in System/390® hexadecimal floating point (HFP) format. (S390) is the format that Db2 stores floating point numbers in. It is also the default value if you do not explicitly specify the FLOAT keyword.

**(IEEE)**

Specifies that LOAD is to expect that floating point numbers are provided in IEEE binary floating point (BFP) format.

When you specify FLOAT(IEEE), Db2 converts the BFP data to HFP format as the data is being loaded into the Db2 table. If a conversion error occurs while Db2 is converting from BFP to HFP, Db2 places the record in the discard file.
FLOAT(IEEE) is mutually exclusive with any specification of the FORMAT keyword. If you specify both FLOAT(IEEE) and FORMAT, Db2 issues message DSNU070I.

BFP format is sometimes called IEEE floating point.

**EBCDIC**
Specifies that the input data file is EBCDIC. The default is EBCDIC.

**ASCII**
Specifies that the input data file is ASCII. Numeric, date, time, and timestamp internal formats are not affected by the ASCII option.

**UNICODE**
Specifies that the input data file is Unicode. The UNICODE option does not affect the numeric internal formats.

**CCSID**
Specifies up to three coded character set identifiers (CCSIDs) for the input file. The first value specifies the CCSID for SBCS data that is found in the input file, the second value specifies the CCSID for mixed DBCS data, and the third value specifies the CCSID for DBCS data. If any of these values is specified as 0 or omitted, the CCSID of the corresponding data type in the input file is assumed to be the same as the installation default CCSID. If the input data is EBCDIC, the omitted CCSIDs are assumed to be the EBCDIC CCSIDs that are specified at installation, and if the input data is ASCII, the omitted CCSIDs are assumed to be the ASCII CCSIDs that are specified at installation. If the CCSIDs of the input data file do not match the CCSIDs of the table that is being loaded, the input data is converted to the table CCSIDs before being loaded.

*integer* is any valid CCSID specification.

If the input data is Unicode, the default CCSID values are the Unicode CCSIDs that are specified at system installation.

**NOSUBS**
Specifies that LOAD is not to accept substitution characters in a string.

Place a substitution character in a string when that string is being converted from ASCII to EBCDIC, or when the string is being converted from one CCSID to another. For example, this substitution occurs when a character (sometimes referred to as a code point) that exists in the source CCSID (code page) does not exist in the target CCSID (code page).

When you specify the NOSUBS option and the LOAD utility determines that a substitution character has been placed in a string as a result of a conversion, it performs one of the following actions:

- **If discard processing is active**: Db2 issues message DSNU310I and places the record in the discard file.
- **If discard processing is not active**: Db2 issues message DSNU334I, and the utility abnormally terminates.

**ENFORCE**
Specifies whether LOAD is to enforce check constraints and referential constraints, except informational referential constraints, which are not enforced.

**CONSTRAINTS**
Indicates that constraints are to be enforced. If LOAD detects a violation, it
deletes the errant row and issues a message to identify it. If you specify this option and referential constraints exist, sort input and sort output data sets must be defined.

**NO** Indicates that constraints are not to be enforced. This option places the target table space in the CHECK-pending status if at least one referential constraint or check constraint is defined for the table.

**NOCHECKPEND** Specifies that LOAD does not set the target table space in the CHECK-pending status if at least one referential constraint or check constraint is defined for the table. This option applies only when ENFORCE NO is specified or enforced by the LOAD utility. A NOCHECKPEND specification does not remove any CHECK-pending status that was set prior to the LOAD.

**ERRDDN** *ddname*

Specifies the DD statement for a work data set that is being used during error processing. Information about errors that are encountered during processing is stored in this data set. A SYSERR data set is required if you request discard processing.

*ddname* is the DD name.

The default value is **SYSERR**.

The ERRDDN keyword specifies either a DD name or a TEMPLATE name specification from a previous TEMPLATE control statement. If utility processing detects that the specified name is both a DD name in the current job step and a TEMPLATE name, the utility uses the DD name. For more information about TEMPLATE specifications, see Chapter 34, “TEMPLATE,” on page 793.

**MAPDDN** *ddname*

Specifies the DD statement for a work data set that is to be used during error processing. The work data set is used to correlate the identifier of a table row with the input record that caused an error. A SYSMAP data set is required if you specify ENFORCE CONSTRAINTS and the tables have a referential relationship, or if you request discard processing when loading one or more tables that contain unique indexes or extended indexes.

*ddname* is the DD name.

The default value is **SYSMAP**.

The MAPDDN keyword specifies either a DD name or a TEMPLATE name specification from a previous TEMPLATE control statement. If utility processing detects that the specified name is both a DD name in the current job step and a TEMPLATE name, the utility uses the DD name. For more information about TEMPLATE specifications, see Chapter 34, “TEMPLATE,” on page 793.

**DISCARDNN** *ddname*

Specifies a DD name or a template name for a discard data set that is to hold copies of records that are not loaded (for example, if they contain conversion errors). The discard data set also holds copies of records that are loaded and then removed (because of unique index errors, referential or check constraint violations, or index evaluation errors). Flag input records for discarding during RELOAD, INDEXVAL, and ENFORCE phases. However, the discard data set is not written until the DISCARD phase when the flagged records are copied from the input data set to the discard data set. The discard data set must be a
sequential data set that can be written to by BSAM, with the same record format, record length, and block size as the input data set.

`ddname` is the DD name.

The default value is **SYSDISC**.

If you omit the DISCARDDN option, the utility application program saves discarded records only if a SYSDISC DD statement is in the JCL input.

The DISCARDDN keyword is not supported if you use a BatchPipes® file as an input to LOAD, using INDDN name for TEMPLATE SUBSYS.

**DISCARDS** *integer*

Specifies the maximum number of source records that are to be written on the discard data set. *integer* can range 0 - 2147483647. If the discard maximum is reached, LOAD abnormally terminates, the discard data set is empty, and you cannot see which records were discarded. You can either restart the job with a larger limit, or terminate the utility.

DISCARDS 0 specifies that you do not want to set a maximum value. The entire input data set can be discarded.

The default value is **0**.

**SORTDEVT** *device-type*

Specifies the device type for temporary data sets that are to be dynamically allocated by the external sort program. You can specify any disk device type that is acceptable to the DYNALLOC parameter of the SORT or OPTION options for the sort program. Tape devices are not supported by the sort program.

If you omit SORTDEVT and a sort is required, you must provide the DD statements that the sort application program needs for the temporary data sets.

A TEMPLATE specification does not dynamically allocate sort work data sets. The SORTDEVT keyword controls dynamic allocation of these data sets.

**SORTNUM** *integer*

Specifies the number of temporary data sets that are to be dynamically allocated by the sort application program.

*integer* is the number of temporary data sets that can range from 2 to 255.

If you omit SORTDEVT, SORTNUM is ignored. If you use SORTDEVT and omit SORTNUM, no value is passed to the sort program. In this case, the sort program uses its own default.

You need at least two sort work data sets for each sort. The SORTNUM value applies to each sort invocation in the utility. For example, if three indexes, SORTKEYS is specified, there are no constraints that limit parallelism, and SORTNUM is specified as 8, a total of 24 sort work data sets are allocated for a job.

Each sort work data set consumes both above-the-line and below-the-line virtual storage, so if you specify a value for SORTNUM that is too high, the utility might decrease the degree of parallelism due to virtual storage constraints, and possibly decreasing the degree down to one, meaning no parallelism.

**Important:** The SORTNUM keyword is ignored if the UTSORTAL subsystem parameter is set to YES and the IGNSORTN subsystem parameter is set to YES.
CONTINUEIF
Indicates that you want to be able to treat each input record as a portion of a larger record. After CONTINUEIF, write a condition in one of the following forms:

\[(start:end) = X'byte-string'\]
\[(start:end) = 'character-string'\]

If the condition is true in any record, the next record is concatenated with it before loading takes place. You can concatenate any number of records into a larger record, up to a maximum size of 32767 bytes.

Character-string constants should be specified in LOAD utility control statements in the character set that matches the input data record. Specify EBCDIC constants in the LOAD control statement if your data is in EBCDIC and specify UNICODE constants if your data is in UNICODE. You may also code the CONTINUEIF condition using the hexadecimal form. For example, use \((1:1)=X'31'\) rather than \((1:1)='1'\).

\[(start:end)\]
Specifies column numbers in the input record; the first column of the record is column 1. The two numbers tell the starting and ending columns of a continuation field in the input record.

Other field position specifications (such as those for WHEN, POSITION, or NULLIF) refer to the field position within the final assembled load record, not within the input record.

The continuation field is removed from the input record and is not part of the final load record.

If you omit :end, Db2 assumes that the length of the continuation field is the length of the byte string or character string. If you use :end, and the length of the resulting continuation field is not the same as the length of the byte string or character string, the shorter string is padded. Character strings are padded with blanks. Hexadecimal strings are padded with zeros.

\[X'byte-string'\]
Specifies a string of hexadecimal characters. This byte-string value in the continuation field indicates that the next input record is a continuation of the current load record. Records with this byte-string value are concatenated until the value in the continuation field changes. For example, the following CONTINUEIF specification indicates that for any input records that have a value of X'FF' in column 72, LOAD is to concatenate that record with the next input record.

\[\text{CONTINUEIF (72) = X'FF'}\]

\['character-string'\]
Specifies a string of characters that has the same effect as \[X'byte-string'\]. For example, the following CONTINUEIF specification indicates that for any input records that have the string CC in columns 99 and 100, LOAD is to concatenate that record with the next input record.

\[\text{CONTINUEIF (99:100) = 'CC'}\]

IGNORE
Specifies that the LOAD utility ignores records that it rejects for the specified reasons. If discarding is specified, no ignored rows are loaded or written to the DISCARD data set. If discarding is not specified, ignored records do not cause the LOAD utility to terminate.
Use the record count messages in the Utility output to determine the number of records ignored.

**WHEN**
Specifies that records that do not satisfy the WHEN clause are ignored.

**PART**
Specifies that records that do not satisfy any partition being loaded are ignored.

**CONV**
Specifies that records that cause a conversion error are ignored.

**VALPROC**
Specifies that records that fail a validation procedure are ignored.

**IDERROR**
Specifies that records that have an identity column value that is out of range are ignored.

**DUPKEY**
Specifies that records that cause a duplicate key error are ignored.

Use the record count messages in the Utility output to determine the number of records ignored.

**DECFLOAT_ROUNDMODE**
Specifies the rounding mode to use when DECFLOATs are manipulated. The following rounding modes are supported:

**ROUND_CEILING**
Round toward +infinity. The discarded digits are removed if they are all zero or if the sign is negative. Otherwise, the result coefficient should be incremented by 1 (rounded up).

**ROUND_DOWN**
Round toward 0 (truncation). The discarded digits are ignored.

**ROUND_FLOOR**
Round toward -infinity. The discarded digits are removed if they are all zero or positive. Otherwise, the sign is negative and the result coefficient should be incremented by 1 (rounded up).

**ROUND_HALF_DOWN**
Round to the nearest number. If equidistant, round down. If the discarded digits are greater than 0.5, the result coefficient should be incremented by 1 (rounded up). The discarded digits are ignored if they are 0.5 or less.

**ROUND_HALF_EVEN**
Round to the nearest number. If equidistant, round so that the final digit is even. If the discarded digits are greater than .05, the result coefficient should be incremented by 1 (rounded up). The discarded digits are ignored if they are less than 0.5. If the result coefficient is .05 and the rightmost digit is even, the result coefficient is not altered. If the result coefficient is .05 and the rightmost digit is odd, the result coefficient should be incremented by 1 (rounded up).

**ROUND_HALF_UP**
Round to nearest. If equidistant, round up. If the discarded digits are greater than or equal to 0.5, the result coefficient should be incremented by 1 (rounded up). Otherwise the discarded digits are ignored.
ROUND_UP
Round away from 0. If all of the discarded digits are 0, the result is unchanged. Otherwise, the result coefficient should be incremented by 1 (rounded up).

If you do not specify DECFLOAT_ROUNDMODE, the LOAD statement uses the DFPDEFDM value in the application defaults load module as the default value. The application defaults load module is either DSNHDECP, which is the default, or a user-specified application defaults load module.

OVERRIDE
Allows unloaded data to be reloaded into the specified types of GENERATED ALWAYS columns.

SYSTEMPERIOD
Allows unloaded data to be reloaded into a GENERATED ALWAYS row-begin or row-end column. Row-begin and row-end columns are intended to be used in the definition of a system period, but the period does not need to exist when the SYSTEMPERIOD keyword is specified.

If you specify OVERRIDE(SYSTEMPERIOD) and include input field specifications in the LOAD statement, both the row-begin and row-end columns that can be used to define a system period must be specified. In the specification for these columns, the NULLIF and DEFAULTIF options are not allowed.

IDENTITY
Allows unloaded data to be reloaded into a GENERATED ALWAYS identity column.

If you specify OVERRIDE(IDENTITY) and include input field specifications in the LOAD statement, the identity column must be specified. In the specification for this column, the NULLIF and DEFAULTIF options are not allowed.

Specify OVERRIDE(IDENTITY) when you want to run LOAD with the INTO TABLE PART clause and a generated identity column is part of the partitioning index. (The generated identity column can be defined as GENERATED ALWAYS or GENERATED BY DEFAULT.)

TRANSID
Allows unloaded data to be reloaded into a GENERATED ALWAYS column that is defined as a transaction-start-ID column.

NONDETERMINISTIC
Allows unloaded data to be reloaded into a GENERATED ALWAYS column that is defined by a non-deterministic expression.

DRAIN_WAIT integer
Specifies the number of seconds that the utility waits when draining the table space or index. The specified time is the aggregate time for objects that are to be reorganized. This value overrides the values that are specified by IRLMRWT and UTIMOUT subsystem parameters. Valid values for integer are from 0 to 1800. If the keyword is omitted or if a value of 0 is specified, the utility uses the value of the IRLMRWT subsystem parameter.

RETRY integer
Specifies the maximum number of retries that LOAD is to attempt to drain. integer is value 0 - 255.

The default value is the value of the UTIMOUT subsystem parameter.
RETRY_DELAY integer

Specifies the minimum duration, in seconds, between retries. integer is a value 1 - 1800.

If you do not specify RETRY_DELAY, LOAD uses the smaller of the following two values:

- DRAIN_WAIT value × RETRY value
- DRAIN_WAIT value × 10

SWITCHTIME

Specifies the time for the drain all in the SWITCH phase to begin. The final result and all of the time stamp calculations of SWITCHTIME are in TIMESTAMP(6). This keyword requires SHRLEVEL REFERENCE. If specified, LOAD sits in the beginning of SWITCH phase and does not attempt to quiesce application access until the specified SWITCHTIME is met.

NONE

Does not specify a time for the drain in the SWITCH phase to begin. This option is the default behavior.

timestamp

Specifies the time that the drain in the SWITCH phase is to begin. This time must not have already occurred when LOAD is run.

labeled-duration-expression

Calculates the time for the drain in the SWITCH phase is to begin. The calculation is based on either CURRENT TIMESTAMP or CURRENT DATE. You can add or subtract one or more constant values to specify the switch time. This switch time must not have already occurred when LOAD is run. CURRENT TIMESTAMP and CURRENT DATE are evaluated a single time when the LOAD statement is first processed. If a list of objects is specified, the same value applies for all objects in the list.

CURRENT_DATE

Specifies that the deadline is to be calculated based on the CURRENT DATE.

CURRENT_TIMESTAMP

Specifies that the deadline is to be calculated based on the CURRENT TIMESTAMP.

WITH_TIMEZONE

 Specifies that the CURRENT TIMESTAMP is compared with the time zone column. The time stamp precision of the special register CURRENT TIMESTAMP should be the same as the column time stamp precision. Otherwise, the default time stamp precision is used. The time zone of CURRENT TIMESTAMP is the value of special register CURRENT TIMEZONE. The comparison is done by comparing the Coordinated Universal Time portion of the time stamp.

constant

Indicates a unit of time and is followed by one of the seven duration keywords: YEARS, MONTHS, DAYS, HOURS, MINUTES, SECONDS, or MICROSECONDS. The singular forms of these words are also accepted: YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, MICROSECOND.
IMPLICIT_TZ

Specifies the implicit time zone to use when the timestamp value that is being loaded does not contain a time zone, and the data type of the target column is TIMESTAMP WITH TIME ZONE.

'timezone-string'

Specifies the implicit time zone value. The time zone is the difference (in hours and minutes) between local time and UTC. The range of the hour component is -12 to 14, and the minute component is 00 to 59. The time zone is specified in the form ±th:tm, with values ranging from -12:59 to +14:00.

If you do not specify the IMPLICIT_TZ option, LOAD uses the value from the IMPLICIT_TIMEZONE DECP value. For more information about this DECP value, see [IMPLICIT TIME ZONE field (IMPLICIT_TIMEZONE DECP value) (Db2 Installation and Migration)](#).

INTO-TABLE-spec

The INTO-TABLE-spec control statement, with its multiple options, defines the function that the utility job performs. More than one table or partition for each table space can be loaded with a single invocation of the LOAD utility. At least one INTO TABLE statement is required for each table that is to be loaded. Each INTO TABLE statement:

- Identifies the table that is to be loaded
- Describes fields within the input record
- Defines the format of the input data set

All tables that are specified by INTO TABLE statements must belong to the same table space.

If the data is already in UNLOAD or SQL/DS format, and FORMAT UNLOAD or FORMAT SQL/DS is used on the LOAD statement, no field specifications are allowed.

Related information:

“Loading partitions” on page 321

When loading XML or LOB columns from a VBS data set, the LOB and XML values need to be at the end of the record as specified by a field specification list.

INTO-TABLE-spec:
INTO TABLE table-name
  IGNOREFIELDS NO
  IGNOREFIELDS YES

PART integer
  PREFORMAT
  resume-spec
  INDDN SYSTEM
  DISCARDN ddname
  INCURSOR cursor-name

NUMRECS integer
  WHEN SQL/DS='table-name'
    field selection criterion
    field specification

resume-spec:

(1)

RESUME NO
  REPLACE
  REUSE
  copy-spec
  RESUME YES
  copy-spec
  KEEPDICTIONARY

Notes:
1 The value specified in the INTO TABLE for RESUME or REPLACE overrides the default value for LOAD RESUME.

field selection criterion:

field specification:
field name spec:

strip spec:

Notes:
1 If you specify GRAPHIC, BINARY, VARBINARY, or VARGRAPHIC, you cannot specify 'strip-char'. You can specify only X'strip-char'.

decimal spec:

Option descriptions for INTO TABLE

table-name
Specifies the name of the table that is to be loaded. The table must be described in the catalog.

The table must not be a catalog table or a system-maintained materialized query table.

If the table name is not qualified by a schema name, the authorization ID of the invoker of the utility job step is used as the schema qualifier of the table name. Enclose the table name in quotation marks if the name contains a blank.

Data from every LOAD record in the data set is loaded into the specified table unless:
- A WHEN clause is used, and the data does not match the field selection criterion.
- The FORMAT UNLOAD option is used on the LOAD statement, and the data comes from a table that is not specified in an INTO TABLE statement.
- A certain partition is specified, and the data does not belong to that partition.
- Data conversion errors occur.
- Any errors occur that are not generated by data conversion.

**IGNOREFIELDS**
Indicates whether LOAD is to skip fields in the input data set that do not correspond to columns in the target table. Examples of fields that do not correspond to table columns are the DSN_NULL_IND_nnnnn, DSN_ROWID, DSN_IDENTITY, and DSN_RCTIMESTAMP fields that are generated by the REORG utility.

**NO** Specifies that the LOAD process is not to skip any fields.

**YES**
Specifies that LOAD is to skip fields in the input data set that do not correspond to columns in the target table.

Specifying YES can be useful if each input record contains a variable-length field, followed by some variable-length data that you do not want to load and then some data that you want to load. Because of the variable-length field, you cannot use the POSITION keyword to skip over the variable-length data that you do not want to load. By specifying IGNOREFIELDS, you can give a field specification for the variable-length data that you do not want to load; and by giving it a name that is not one of the table column names, LOAD skips the field without loading it.

Use this option with care, because it also causes fields to be skipped if you intend to load a column but have misspelled the name.

**NUMRECS**
Indicates the number of input records for the specified table or table partition.

**integer**
A positive integer that is used as an estimate of the number of complete input records that are to be loaded into the specified table. The specified number refers to fully assembled input records when CONTINUEIF is used.

Use the NUMRECS keyword for multi-table table spaces to indicate the number of input records that will be loaded into each of the tables or table partitions.

Specifying the number of records improves the sizing of the sort work data sets that the utility requires when indexes are built in parallel. If the LOAD utility underestimates the size of the sort work data sets, the execution of the LOAD utility could fail.

You can also use the NUMRECS keyword when the input data set is located on tape or if only a fraction of the input records will be loaded.

If an integer value is specified on the SORTKEYS keyword at the table-space level, the NUMRECS keyword cannot be specified in the same LOAD statement.
If multiple tables or partitions are loaded in the same LOAD statement, the NUMRECS keyword must be specified either for all of the tables or partitions or for none of the tables or partitions.

**PART** integer

Specifies that data is to be loaded into a partition of a partitioned table space. This option is valid only for partitioned table spaces, not including partition-by-growth table spaces.

integer is the physical partition number for the partition into which records are to be loaded. The same partition number cannot be specified more than once if partition parallelism has been requested. Any data that is outside the range of the specified partition is not loaded. The maximum is 4096.

LOAD INTO PART integer is not allowed if:

- An identity column is part of the partitioning index, unless OVERRIDE(IDENTITY) is specified for the identity column GENERATED ALWAYS
- A row ID is part of the partitioning index
- The table space is partition-by-growth

For nonpartitioned secondary indexes, LOAD PART:
- Does not set the page set REBUILD-pending (PSRBD) status
- Does not consider PCTFREE or FREEPAGE attributes when inserting keys

**PREFORMAT**

Specifies that the remaining pages are to be preformatted up to the high-allocated RBA in the partition and its corresponding partitioning index space. The preformatting occurs after the data is loaded and the indexes are built.

**INDEXDEFER**

Specifies whether index builds are done during the BUILD phase of LOAD, or are deferred until REBUILD INDEX is run manually. Deferring index builds is a way to improve LOAD performance, especially for LOAD with PART. If indexes are not built during LOAD, LOAD places the affected indexes in the REBUILD-pending state.

- **NONE** Specifies that indexes are built during the BUILD phase of LOAD.
- **ALL** Specifies that no indexes are built as part of a BUILD phase of the LOAD utility. Index builds are deferred until REBUILD INDEX is run manually. ALL is valid only if SHRLEVEL NONE is also specified.
- **NPI** Specifies that building of nonpartitioned indexes is not done as part of a BUILD phase of the LOAD utility. Nonpartitioned index builds are deferred until REBUILD INDEX is run manually. NPI is valid only if SHRLEVEL NONE is also specified.
- **NONUNIQUE** Specifies that building of only nonunique indexes is deferred. NONUNIQUE is valid only if ALL or NPI is also specified. If NONUNIQUE is not specified, building of unique and nonunique indexes is deferred. If unique indexes are defined on the tables that are being loaded, specify NONUNIQUE unless the data really is unique. REBUILD INDEX does not resolve duplicate keys for unique indexes.
When INDEXDEFER ALL or INDEXDEFER NPI is specified:

- If ENFORCE CONSTRAINTS is also specified, building of indexed foreign keys is not deferred.
- If RESUME is also specified, building of indexes that were created with DEFINE NO and are still undefined is not deferred. Building of undefined indexes is deferred only when REPLACE is specified.

Any LOAD REPLACE SHRLEVEL REFERENCE PART execution results in the default of INDEXDEFER NPI, with the affected part-level non-partitioning index placed put into RBDP status on successful LOAD completion.

NONUNIQUE is ignored if specified.

RESUME

Specifies whether records are to be loaded into an empty or non-empty partition. For nonsegmented table spaces, space is not reused for rows that have been marked as deleted or by rows of dropped tables is not reused. If the RESUME option is specified at the table space level, the RESUME option is not allowed in the PART clause.

If you want the RESUME option to apply to the entire table space, use the LOAD RESUME option. If you want the RESUME option to apply to a particular partition, specify it by using PART integer RESUME.

NO Loads records into an empty partition. If the partition is not empty, and you have not used REPLACE, a message is issued, and the utility job step terminates with a job step condition code of 8.

For non-segmented table spaces that contains deleted rows or rows of dropped tables, using the REPLACE keyword provides increased efficiency.

YES Loads records into a non-empty partition. If the partition is empty, a warning message is issued, but the partition is loaded.

If COPYDDN or RECOVERDDN is specified with RESUME YES, an inline image copy will be created for the specified partition during processing of the LOAD statement. The image copy is created at the table space level and is a full image copy for each partition that is specified by INTO TABLE PART. If COPYDDN or RECOVERDDN was specified on the table space level, it cannot be specified at the INTO TABLE PART level.

REPLACE

Indicates that you want to replace only the contents of the partition that is cited by the PART option, rather than the entire table space.

You cannot use LOAD REPLACE with the PART integer REPLACE option of INTO TABLE. If you specify the REPLACE option, you must either replace an entire table space, using LOAD REPLACE, or a single partition, using the PART integer REPLACE option of INTO TABLE. You can, however, use PART integer REPLACE with LOAD RESUME YES.

REUSE

Specifies, when used with the REPLACE option, that LOAD should logically reset and reuse Db2-managed data sets without deleting and redefining them.

If you do not specify REUSE, Db2 deletes and redefines Db2-managed data sets to reset them.

If you specify REUSE with REPLACE on the PART specification (and not for LOAD at the table space level), only the specified partitions are logically reset. If you specify REUSE for the table space and REPLACE for the partition, data sets for the replaced parts are logically reset.
**KEEPDICTIONARY**

Specifies that the LOAD utility is not to build a new dictionary. LOAD retains the current dictionary and uses it for compressing the input data. This option eliminates the cost that is associated with building a new dictionary.

This keyword is valid only if a dictionary exists and the partition that is being loaded has the COMPRESS YES attribute.

If the partition has the COMPRESS YES attribute, but no dictionary exists, one is built and an error message is issued.

**INDDN** *ddname*

Specifies the input data set for the partition.

*ddname* is the name of a DD statement from the job step or the name of a template in the utility input stream. The default value is **SYSREC**.

The record format for an input data set must be fixed-length or variable-length. The data set must be a sequential data set that is readable by the basic sequential access method (BSAM).

When loading LOB data by using file reference variables, the input data set should include the names of the files that contain the LOB column values. Each file can be either a sequential file, PDS member, PDSE member, or separate HFS file.

If you specify INDDN in one INTO TABLE PART clause, you must specify INDDN in all other INTO TABLE PART clauses in that LOAD statement.

Providing input data sets at the partition level (by specifying multiple INTO TABLE PART INDDN clauses) enables LOAD partition parallelism, which can significantly improve performance. When one or more nonpartitioned secondary indexes exist on the table space, loading all partitions in a single job with partition parallelism is recommended instead of running concurrent separate jobs.

The field specifications apply separately to each input file. Therefore, if multiple INTO TABLE PART INDDN clauses are used, field specifications are required for each one.

**DISCARDDN** *ddname*

Specifies the DD statement for a discard data set for the partition. The discard data set holds copies of records that are not loaded (for example, if they contain conversion errors). The discard data set also holds copies of records that were loaded and then removed (due to unique index errors, or referential or check constraint violations).

If DISCARDS n is specified in the LOAD statement and LOAD partition parallelism is enabled, specifying DISCARDDN in an INTO TABLE PART clause is recommended. See “Loading partitions” on page 321.

Flag input records for discarding during the RELOAD, INDEXVAL, and ENFORCE phases. However, the utility does not write the discard data set until the DISCARD phase when the utility copies the flagged records from the input data set to the discard data set.

The discard data set must be a sequential data set, and it must be write-accessible by BSAM, with the same record format, record length, and block size as the input data set.

The *ddname* is the name of the discard data set. DISCARDDN can be a template name.
If you omit the DISCARDDN option, LOAD does not save discarded records.

**INCURSOR cursor-name**

Specifies the cursor for the input data set. You must declare the cursor before it is used by the LOAD utility. Use the EXEC SQL utility control statement to define the cursor. You cannot load data into the same table on which you defined the cursor.

The specified cursor can be used as part of the Db2 family cross loader function, which enables you to load data from any DRDA-compliant remote server. For more information about using the cross loader function, see "Loading data by using the cross-loader function" on page 333.

`cursor-name` is the cursor name. Cursor names that are specified with the LOAD utility cannot be longer than eight characters.

You cannot use the INCURSOR option with the following options:

- SHRLEVEL CHANGE
- NOSUBS
- FORMAT UNLOAD
- FORMAT SQL/DS
- CONTINUEIF
- WHEN
- SPANNED YES

In addition, you cannot specify field specifications with the INCURSOR option.

**WHEN**

Indicates which records in the input data set are to be loaded. If no WHEN clause is specified (and if FORMAT UNLOAD was not used in the LOAD statement), all records in the input data set are loaded into the specified tables or partitions. (Data that is beyond the range of the specified partition is not loaded.)

The option following WHEN describes a condition; input records that satisfy the condition are loaded. Input records that do not satisfy any WHEN clause of any INTO TABLE statement are written to the discard data set, if one is being used and the IGNORE(WHEN) option is not specified.

Character-string constants should be specified in LOAD utility control statements in the character set that matches the input data record. Specify EBCDIC constants in the LOAD control statement if your data is in EBCDIC and specify UNICODE constants if your data is in UNICODE. You may also code the WHEN condition using the hexadecimal form. For example, use \( (1:1)=X'31' \) rather than \( (1:1)='1' \).

**SQL/DS='table-name'**

Is valid only when the FORMAT SQL/DS option is used on the LOAD statement.

`table-name` is the name of a table that has been unloaded into the unload data set. The table name after INTO TABLE tells which Db2 table the SQL/DS table is loaded into. Enclose the table name in quotation marks if the name contains a blank.

If no WHEN clause is specified, input records from every SQL/DS table are loaded into the table that is specified after INTO TABLE.

**field-selection-criterion**

Describes a field and a character constant. Only those records in which the field contains the specified constant are to be loaded into the table that is specified after INTO TABLE.
A field in a selection criterion must:

- Contain a character or graphic string. No data type conversions are performed when the contents of the field in the input record are compared to a string constant.
- Start at the same byte offset in each assembled input record. If any record contains varying-length strings, which are stored with length fields, that precede the selection field, they must be padded so that the start of the selection field is always at the same offset.

The field and the constant do not need to be the same length. If they are not, the shorter of the two is padded before a comparison is made. Character and graphic strings are padded with blanks. Hexadecimal strings are padded with zeros.

### field-name
Specifies the name of a field that is defined by a field-specification. If field-name is used, the start and end positions of the field are given by the POSITION option of the field specification.

### (start:end)
Identifies column numbers in the assembled load record; the first column of the record is column 1. The two numbers indicate the starting and ending columns of a selection field in the load record.

If :end is not used, the field is assumed to have the same length as the constant.

### X'byte-string'
Identifies the constant as a string of hexadecimal characters. For example, the following WHEN clause specifies that a record is to be loaded if it has the value X'FFFF' in columns 33 through 34.

WHEN (33:34) = X'FFFF'

### 'character-string'
Identifies the constant as a string of characters. For example, the following WHEN clause specifies that a record is to be loaded if the field DEPTNO has the value D11.

WHEN DEPTNO = 'D11'

If the field uses a specific external date, time, or timestamp format, the field specification must use a character string that matches the specified format.

### G'graphic-string'
Identifies the constant as a string of double-byte characters. For example, the following WHEN clause specifies that a record is to be loaded if it has the specified value in columns 33 through 36.

WHEN (33:36) = G'<<*>>'

In this example, < is the shift-out character, * is a double-byte character, and > is the shift-in character.

If the first or last byte of the input data is a shift-out character, it is ignored in the comparison. Specify G as an uppercase character.

### N'graphic-string'
Identifies the constant as a string of double-byte characters. N and G are synonymous for specifying graphic string constants. Specify N as an uppercase character.
(field-specification, ...)  
Describes the location, format, and null value identifier of the data that is to be loaded.

If no field specifications are used:
- The fields in the input records are assumed to be in the same order as in the Db2 table.
- The formats are set by the FORMAT option on the LOAD statement, if that option is used.
- Fixed strings in the input are assumed to be of fixed maximum length. VARCHAR and VARGRAPHIC fields must contain a valid 2-byte binary length field preceding the data; no intervening gaps are allowed between the VARCHAR or VARGRAPHIC fields and the field that follows.
- BINARY fields are assumed to be of fixed maximum length.
- VARBINARY fields must contain a valid 2-byte binary length field preceding the data.
- ROWID fields are varying length, and must contain a valid 2-byte binary length field preceding the data; no intervening gaps are allowed between ROWID fields and the fields that follow.
- LOB fields are varying length, and require a valid 4-byte binary length field preceding the data; no intervening gaps are allowed between them and the LOB fields that follow.
- Numeric data is assumed to be in the appropriate internal Db2 number representation.
- The NULLIF or DEFAULTIF options cannot be used.

If any field specification is used for an input table, a field specification must exist for each field of the table that does not have a default value. Any field in the table with no corresponding field specification is loaded with its default value.

If any column in the output table does not have a field specification and is defined as NOT NULL, with no default, the utility job step is terminated.

Identity columns or row change timestamp columns can appear in the field specification only if you defined them with the GENERATED BY DEFAULT attribute.

If you are loading application or system temporal data and you include field specifications, you must specify both the start and end time column fields.

field-name
Specifies the name of a field, which can be a name of your choice. If the field is to be loaded, the name must be the name of a column in the table that is named after INTO TABLE unless IGNOREFIELDS is specified. You can use the field name as a vehicle to specify the range of incoming data. See Example 4: [Loading data of different data types] for an example of loading selected records into an empty table space.

The starting location of the field is given by the POSITION option. If POSITION is not used, the starting location is one column after the end of the previous field.

LOAD determines the length of the field in one of the following ways, in the order listed:
1. If the field has data type VARCHAR, VARGRAPHIC, VARBINARY, ROWID, or XML the length is assumed to be contained in a 2-byte binary
field that precedes the data. For VARCHAR, VARBINARY, and XML fields, the length is in bytes; for VARGRAPHIC fields, the length field identifies the number of double-byte characters.

If the field has data type CLOB, BLOB, or DBCLOB, the length is assumed to be contained in a 4-byte binary field that precedes the data. For BLOB and CLOB fields, the length is in bytes; for DBCLOB fields, the length field identifies the number of double-byte characters.

2. If \texttt{end} is used in the \texttt{POSITION} option, the length is calculated from \texttt{start} and \texttt{end}. In that case, any length attribute after the CHAR, GRAPHIC, INTEGER, DECIMAL, FLOAT, or DECFLOAT specifications is ignored.

3. The length attribute on the CHAR, GRAPHIC, INTEGER, DECIMAL, FLOAT, or DECFLOAT specifications is used as the length.

4. The length is taken from the Db2 field description in the table definition, or it is assigned a default value according to the data type. For DATE and TIME fields, the length is defined during installation. For variable-length fields, the length is defined from the column in the Db2 table definition, excluding the null indicator byte, if it is present. The following table shows the default length, in bytes, for each data type.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|}
\hline
\textbf{Data type} & \textbf{Default length in bytes} \\
\hline
BIGINT & 8 \\
BINARY & Length that is used in column definition \\
BLOB & Varying \\
CHARACTER & Length that is used in column definition \\
CLOB & Varying \\
DATE & 10 (or installation default) \\
DBCLOB & Varying \\
DECFLOAT(16) & 8 \\
DECFLOAT(34) & 16 \\
DECIMAL EXTERNAL & Decimal precision for output columns that are decimal, otherwise the length that is used in column definition \\
DECIMAL PACKED & Length that is used in column definition \\
DECIMAL ZONED & Decimal precision for output columns that are decimal, otherwise the length that is used in column definition \\
FLOAT (single precision) & 4 \\
FLOAT (double precision) & 8 \\
GRAPHIC & 2 multiplied by (length that is used in column definition) \\
INTEGER & 4 \\
MIXED & Mixed DBCS data \\
ROWID & Varying \\
SMALLINT & 2 \\
TIME & 8 (or installation default) \\
TIMESTAMP & 26 \\
VARBINARY & Varying \\
\hline
\end{tabular}
\caption{Default length of each data type (in bytes)}
\end{table}
Table 29. Default length of each data type (in bytes) (continued)

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default length in bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARCHAR</td>
<td>Varying</td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td>Varying</td>
</tr>
<tr>
<td>XML</td>
<td>Varying</td>
</tr>
<tr>
<td>TIMESTAMP WITH TIME ZONE</td>
<td>33</td>
</tr>
</tbody>
</table>

If a data type is not given for a field, its data type is assumed to be the same as that of the column into which it is loaded, as given in the Db2 table definition.

**POSITION(start:end)**

Indicates where a field is in the assembled load record.

*start* and *end* are the locations of the first and last columns of the field; the first column of the record is column 1. The option can be omitted.

Column locations can be specified as:
- An integer *n*, meaning an actual column number
- *, meaning one column after the end of the previous field
- *+n*, where *n* is an integer, meaning *n* columns after the location that is specified by *

Do not enclose the entire POSITION option specification in parentheses; enclose only the *start:end* description in parentheses. Valid and invalid specifications are shown in the following table.

Table 30. Example of valid and invalid POSITION specifications

<table>
<thead>
<tr>
<th>Valid</th>
<th>Invalid</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSITION (10:20)</td>
<td>(POSITION (10:20))</td>
</tr>
</tbody>
</table>

**CONSTANTIF field-selection-criterion**

Describes a condition that causes the Db2 column to be loaded with a constant value. You can write the field-selection-criterion with the same options as described for field-selection-criterion. If the contents of the CONSTANTIF field match the provided character constant, the field that is specified in field-specification is loaded with the value specified in the CONSTANT keyword.

If the CONSTANTIF field is defined by the name of a VARCHAR or VARGRAPHIC field, Db2 takes the length of the field from the 2-byte binary field that appears before the data portion of the VARCHAR or VARGRAPHIC field.

Character-string constants should be specified in LOAD utility control statements in the character set that matches the input data record. Specify EBCDIC constants in the LOAD control statement if your data is in EBCDIC and specify UNICODE constants if your data is in UNICODE. You may also code the CONSTANTIF condition using the hexadecimal form. For example, if the input data is in EBCDIC and the control statement is in UTF-8, use \((1:1)='31'\) in the condition rather than \((1:1)='1'\).

**CONSTANT(...)**

Specifies that the column that matches the field name should be loaded with a constant value. The field name that is associated with the CONSTANT
keyword must match a column name of the specified table. The length and
type of the CONSTANT field is derived from the specified value and must be
valid for the target column.

For a CONSTANT field, no other field specification options are allowed. A
CONSTANT field cannot be specified for LOB or XML columns.

If a record is discarded because of an invalid value specified with the
CONSTANT keyword, the record in the discard data set reflects the original
record from SYSREC, not the record as loaded with the constant value(s).

'string'
    Specifies the character string that is to be inserted in the target column. If
    the specified string is in the form 'string', it is assumed to be in the
    encoding scheme of SYSIN even if the encoding scheme of SYSREC is not
    the same. Conversion might be applied before inserting the string into the
target column.

'X'hex-string'
    Specifies the hexadecimal value that is to be inserted in the target column.
    The hexadecimal value must be specified in the format 'X'hex-string'.

integer
    Specifies the integer that is to be inserted into the target column. Valid
    values are between -2147483648 and 2147483647.

CURRENT DATE
    Specifies that the current date is to be inserted into the target column.

CURRENT TIME
    Specifies that the current time is to be inserted into the target column.

CURRENT TIMESTAMP
    Specifies that the current timestamp is to be inserted into the target
column.

NULL
    Specifies that the target column should be set to NULL. The target column
    must be nullable.

Data types in a field specification: The data type of the field can be specified by
any of the keywords that follow. Except for graphic fields, length is the length in
bytes of the input field.

All numbers that are designated EXTERNAL are in the same format in the input
records.

CHAR(length)
    Specifies a fixed-length character string. If you do not specify length, the length
    of the string is determined from the POSITION specification. If you do not
    specify length or POSITION, LOAD uses the default length for CHAR, which is
determined from the length of the column in the table. You can also specify
    CHARACTER and CHARACTER(length).

When you specify CHAR as the type for the file name for CLOBF, BLOBF, or
DBCLOBF, you must also provide the length so that the LOAD utility can
determine the correct file name. Otherwise message DSNU338I will be issued
for an invalid column specification.

BIT
    Specifies that the input field contains BIT data. If BIT is specified, LOAD
bypasses any CCSID conversions for the input data. If the target column has the BIT data type attribute, LOAD bypasses any code page translation for the input data.

**MIXED**
Specifies that the input field contains mixed SBCS and DBCS data. If MIXED is specified, any required CCSID conversions use the mixed CCSID for the input data. If MIXED is not specified, any such conversions use the SBCS CCSID for the input data.

**BLOBF**
Indicates that the input field contains the name of a BLOB file which is going to be loaded to a specified BLOB/XML column.

**CLOBF**
Indicates that the input field contains the name of a CLOB file which is going to be loaded to a specified CLOB/XML column.

**DBCLOBF**
Indicates that the input field contains the name of a DBCLOBF file which is going to be loaded to a specified DBCLOB/XML column.

**PRESERVE WHITESPACE**
Specifies that the white space in the XML column is preserved. The default is not to preserve the white space.

**STRIP**
Specifies that LOAD is to remove zeros (the default) or the specified characters from the beginning, the end, or both ends of the data. LOAD pads the CHAR field, so that it fills the rest of the column.

LOAD applies the strip operation before performing any character code conversion or padding.

The effect of the STRIP option is the same as the SQL STRIP scalar function.

**BOTH**
Indicates that LOAD is to remove occurrences of blank or the specified strip character from the beginning and end of the data.

**TRAILING**
Indicates that LOAD is to remove occurrences of blank or the specified strip character from the end of the data.

**LEADING**
Indicates that LOAD is to remove occurrences of blank or the specified strip character from the beginning of the data.

'STRIP-char'
Specifies a single-byte or double-byte character that LOAD is to strip from the data.

Specify this character value in EBCDIC. Depending on the input encoding scheme, LOAD applies SBCS CCSID conversion to the strip-char value before it is used in the strip operation.

If the subtype of the column to be loaded is BIT or you want to specify a strip-char value in an encoding scheme other than EBCDIC, use the
hexadecimal form (X'\text{strip-char}')). LOAD does not perform any CCSID conversion if the hexadecimal form is used.

\textit{X'\text{strip-char}'

Specifies in hexadecimal form a single-byte or double-byte character that LOAD is to strip from the data. For single-byte characters, specify this value in the form X'hh', where \textit{hh} is two hexadecimal characters. For double-byte characters, specify this value in the form X'hhhh', where \textit{hhhh} is four hexadecimal characters.

Use the hexadecimal form to specify a character in an encoding scheme other than EBCDIC. When you specify the character value in hexadecimal form, LOAD does not perform any CCSID conversion.

If you specify a strip character in the hexadecimal format, you must specify the character in the input encoding scheme.

\textbf{TRUNCATE}

Indicates that LOAD is to truncate the input character string from the right if the string does not fit in the target column. LOAD performs the truncation operation after any CCSID translation.

If the input data is BIT data, LOAD truncates the data at a byte boundary. If the input data is SBCS or MIXED data, LOAD truncates the data at a character boundary. (Double-byte characters are not split.) If a MIXED field is truncated to fit a column, the truncated string can be shorter than the specified column size. In this case, blanks in the output CCSID are padded to the right. If MIXED data is in EBCDIC, truncation preserves the SO (shift-out) and SI (shift-in) characters around a DBCS string.

\textbf{VARCHAR}

Specifies a character field of varying length. The length in bytes must be specified in a 2-byte binary field preceding the data. (The length does not include the 2-byte field itself.) The length field must start in the column that is specified as \textit{start} in the POSITION option. If \textit{:end} is used, it is ignored.

\textbf{BIT}

Specifies that the input field contains BIT data. If BIT is specified, LOAD bypasses any CCSID conversions for the input data. If the target column has the BIT data type attribute, LOAD bypasses any code page translation for the input data.

\textbf{MIXED}

Specifies that the input field contains mixed DBCS data. If MIXED is specified, any required CCSID conversions use the mixed CCSID for the input data. If MIXED is not specified, any such conversions use the SBCS CCSID for the input data.

\textbf{BLOBF}

Indicates that the input field contains the name of a BLOB file which is going to be loaded to a specified BLOB/XML column.

\textbf{CLOBF}

Indicates that the input field contains the name of a CLOB file which is going to be loaded to a specified CLOB/XML column.

\textbf{DBCLOBF}

Indicates that the input field contains the name of a DBCLOBF file which is going to be loaded to a specified DBCLOB/XML column.
PRESERVE WHITESPACE
  Specifies that the white space in the XML column is preserved. The default is not to preserve the white space.

STRIP
  Specifies that LOAD is to remove zeros (the default) or the specified characters from the beginning, the end, or both ends of the data. LOAD adjusts the VARCHAR length field to the length of the stripped data.

  LOAD applies the strip operation before performing any character code conversion or padding.

  The effect of the STRIP option is the same as the SQL STRIP scalar function.

BOTH
  Indicates that LOAD is to remove occurrences of blank or the specified strip character from the beginning and end of the data.

TRAILING
  Indicates that LOAD is to remove occurrences of blank or the specified strip character from the end of the data.

LEADING
  Indicates that LOAD is to remove occurrences of blank or the specified strip character from the beginning of the data.

'strip-char'
  Specifies a single-byte or double-byte character that LOAD is to strip from the data.

  Specify this character value in EBCDIC. Depending on the input encoding scheme, LOAD applies SBCS CCSID conversion to the strip-char value before it is used in the strip operation.

  If the subtype of the column to be loaded is BIT or you want to specify a strip-char value in an encoding scheme other than EBCDIC, use the hexadecimal form (X'strip-char'). LOAD does not perform any CCSID conversion if the hexadecimal form is used.

X'strip-char'
  Specifies in hexadecimal form a single-byte or double-byte character that LOAD is to strip from the data. For single-byte characters, specify this value in the form Xhh, where hh is two hexadecimal characters. For double-byte characters, specify this value in the form Xhhhh, where hhhh is four hexadecimal characters.

  Use the hexadecimal form to specify a character in an encoding scheme other than EBCDIC. When you specify the character value in hexadecimal form, LOAD does not perform any CCSID conversion.

  If you specify a strip character in the hexadecimal format, you must specify the character in the input encoding scheme.

TRUNCATE
  Indicates that LOAD is to truncate the input character string from the right if the string does not fit in the target column. LOAD performs the truncation operation after any CCSID translation.

  If the input data is BIT data, LOAD truncates the data at a byte boundary.

  If the input data is character type data, LOAD truncates the data at a character boundary. If a mixed-character type data is truncated to fit a
column of fixed size, the truncated string can be shorter than the specified column size. In this case, blanks in the output CCSID are padded to the right.

**GRAPHIC**(length)
Specifies a fixed-length graphic type. You can specify both start and end for the field specification.

If you use GRAPHIC, the input data must not contain shift characters. start and end must indicate the starting and ending positions of the data itself.

length is the number of double-byte characters. The length of the field in bytes is twice the value of length. If you do not specify length, the number of double-byte characters is determined from the POSITION specification. If you do not specify length or POSITION, LOAD uses the default length for GRAPHIC, which is determined from the length of the column in the table.

For example, let *** represent three double-byte characters. Then, to describe ***, specify either POS(1:6) GRAPHIC or POS(1) GRAPHIC(3). A GRAPHIC field that is described in this way cannot be specified in a field selection criterion.

**STRIP**
Specifies that LOAD is to remove zeros (the default) or the specified characters from the beginning, the end, or both ends of the data.

LOAD applies the strip operation before performing any character code conversion or padding.

The effect of the STRIP option is the same as the SQL STRIP scalar function.

**BOTH**
Indicates that LOAD is to remove occurrences of blank or the specified strip character from the beginning and end of the data.

**TRAILING**
Indicates that LOAD is to remove occurrences of blank or the specified strip character from the end of the data.

**LEADING**
Indicates that LOAD is to remove occurrences of blank or the specified strip character from the beginning of the data.

X'strip-char'
Specifies the hexadecimal form of the double-byte character that LOAD is to strip from the data. Specify this value in the form X'hhhh', where hhhh is four hexadecimal characters.

You must specify the character in the input encoding scheme.

**TRUNCATE**
Indicates that LOAD is to truncate the input character string from the right if the string does not fit in the target column. LOAD performs the truncation operation after any CCSID translation.

LOAD truncates the data at a character boundary. Double-byte characters are not split.

**GRAPHIC EXTERNAL**(length)
Specifies a fixed-length field of the graphic type with the external format. You can specify both start and end for the field specification.

If you use GRAPHIC EXTERNAL, the input data must contain a shift-out character in the starting position, and a shift-in character in the ending
position. Other than the shift characters, this field must have an even number of bytes. The first byte of any pair must not be a shift character.

*length* is the number of double-byte characters. *length* for GRAPHIC EXTERNAL does not include the number of bytes that are represented by shift characters. The length of the field in bytes is twice the value of *length*. If you do not specify *length*, the number of double-byte characters is determined from the POSITION specification. If you do not specify *length* or POSITION, LOAD uses the default length for GRAPHIC, which is determined from the length of the column in the table.

For example, let *** represent three double-byte characters, and let < and > represent shift-out and shift-in characters. Then, to describe <***>, specify either POS(1:8) GRAPHIC EXTERNAL or POS(1) GRAPHIC EXTERNAL(3).

**STRIP**

Specifies that LOAD is to remove zeros (the default) or the specified characters from the beginning, the end, or both ends of the data.

LOAD applies the strip operation before performing any character code conversion or padding.

The effect of the STRIP option is the same as the SQL STRIP scalar function.

**BOTH**

Indicates that LOAD is to remove occurrences of blank or the specified strip character from the beginning and end of the data.

**TRAILING**

Indicates that LOAD is to remove occurrences of blank or the specified strip character from the end of the data.

**LEADING**

Indicates that LOAD is to remove occurrences of blank or the specified strip character from the beginning of the data.

*X*’*strip-char’*

Specifies the hexadecimal form of the double-byte character that LOAD is to strip from the data. Specify this value in the form X’hhhh’, where hhhh is four hexadecimal characters.

You must specify the character in the input encoding scheme.

**TRUNCATE**

Indicates that LOAD is to truncate the input character string from the right if the string does not fit in the target column. LOAD performs the truncation operation after any CCSID translation.

LOAD truncates the data at a character boundary. Double-byte characters are not split.

**VARGRAPHIC**

Identifies a graphic field of varying length. The length, in double-byte characters, must be specified in a 2-byte binary field preceding the data. (The length does not include the 2-byte field itself.) The length field must start in the column that is specified as *start* in the POSITION option. *end*, if used, is ignored.

VARGRAPHIC input data must not contain shift characters.

**STRIP**

Specifies that LOAD is to remove zeros (the default) or the specified
characters from the beginning, the end, or both ends of the data. LOAD adjusts the VARGRAPHIC length field to the length of the stripped data (the number of DBCS characters).

LOAD applies the strip operation before performing any character code conversion or padding.

The effect of the STRIP option is the same as the SQL STRIP scalar function.

**BOTH**
Indicates that LOAD is to remove occurrences of blank or the specified strip character from the beginning and end of the data.

**TRAILING**
Indicates that LOAD is to remove occurrences of blank or the specified strip character from the end of the data.

**LEADING**
Indicates that LOAD is to remove occurrences of blank or the specified strip character from the beginning of the data.

*X'strip-char'*
Specifies the hexadecimal form of the double-byte character that LOAD is to strip from the data. Specify this value in the form X'hhhh', where hhhh is four hexadecimal characters.

You must specify the character in the input encoding scheme.

**TRUNCATE**
Indicates that LOAD is to truncate the input character string from the right if the string does not fit in the target column. LOAD performs the truncation operation after any CCSID translation.

LOAD truncates the data at a character boundary. Double-byte characters are not split.

**SMALLINT**
Specifies a 2-byte binary number. Negative numbers are in two's complement notation.

**INTEGER**
Specifies a 4-byte binary number. Negative numbers are in two's complement notation. You can also specify INT.

**INTEGER EXTERNAL**(length)
A string of characters that represent a number. The format is that of an SQL numeric constant. If you do not specify length, the length of the string is determined from the POSITION specification. If you do not specify length or POSITION, LOAD uses the default length for INTEGER, which is 4 bytes. You can also specify INT EXTERNAL.

**BIGINT**
Specifies an 8-byte binary number. Negative numbers are in two's complement notation.

**BINARY**(length)
Specifies a fixed-length binary string. If you do not specify length, the length of the string is determined from the POSITION specification. If you do not specify length or POSITION, LOAD uses the default length for BINARY, which is determined from the length of the column in the table. The default for X'strip-char' is hexadecimal zero (X'00'). No data conversion is applied to the field.
**STRIP**

Specifies that LOAD is to remove binary zeros (the default) or the specified `X'strip-char'` from the beginning, the end, or both ends of the data. LOAD pads the BINARY field, so that it fills the rest of the column.

The effect of the STRIP option is the same as the SQL STRIP scalar function.

**BOTH**

Indicates that LOAD is to remove occurrences of binary zeros or the specified strip character from the beginning and end of the data.

**TRAILING**

Indicates that LOAD is to remove occurrences of blank or the specified strip character from the end of the data.

**LEADING**

Indicates that LOAD is to remove occurrences of blank or the specified strip character from the beginning of the data.

`X'strip-char'`

Specifies, in hexadecimal form, a single-byte or double-byte character that LOAD is to strip from the data. For single-byte characters, specify this value in the form `X'hh'`, where `hh` is two hexadecimal characters.

**TRUNCATE**

Indicates that LOAD is to truncate the input character string from the right if the string does not fit in the target column.

LOAD truncates the data at a character boundary.

**VARBINARY**

Specifies a varying length binary string. The length in bytes must be specified in a 2-byte binary field preceding the data (the length does not include the 2-byte field itself). The length field must start in the column that is specified as start in the POSITION option. If `:end` is used, it is ignored. The default for `X'strip-char'` is hexadecimal zero (X'00'). No data conversion is applied to the field.

**STRIP**

Specifies that LOAD is to remove binary zeros (the default) or the specified characters from the beginning, the end, or both ends of the data. LOAD pads the VARBINARY field, so that it fills the rest of the column.

The effect of the STRIP option is the same as the SQL STRIP scalar function.

**BOTH**

Indicates that LOAD is to remove occurrences of binary zeros or the specified strip character from the beginning and end of the data.

**TRAILING**

Indicates that LOAD is to remove occurrences of binary zeros or the specified strip character from the end of the data.

**LEADING**

Indicates that LOAD is to remove occurrences of binary zeros or the specified strip character from the beginning of the data.

`X'strip-char'`

Specifies, in hexadecimal form, a single-byte character that LOAD is to strip from the data. For single-byte characters, specify this value in the form `X'hh'`, where `hh` is two hexadecimal characters.
TRUNCATE
Indicates that LOAD is to truncate the input character string from the right if the string does not fit in the target column.
LOAD truncates the data at a character boundary.

DECIMAL PACKED
Specifies a number of the form $d_1d_2...d_sd_1s$, where $d$ is a decimal digit that is represented by four bits, and $s$ is a 4-bit sign value. The plus sign (+) is represented by A, C, E, or F, and the minus sign (-) is represented by B or D. The maximum number of $d$s is the same as the maximum number of digits that are allowed in the SQL definition. You can also specify DECIMAL, DEC, or DEC PACKED.

DECIMAL ZONED
Specifies a number in the form $znznzn...zn/sn$, where $z$, $n$, and $s$ have the following values:

- $n$: A decimal digit represented by the right 4 bits of a byte (called the numeric bits)
- $z$: That digit's zone, represented by the left 4 bits
- $s$: The right-most byte of the decimal operand; $s$ can be treated as a zone or as the sign value for that digit

The plus sign (+) is represented by A, C, E, or F, and the minus sign (-) is represented by B or D. The maximum number of $zn$s is the same as the maximum number of digits that are allowed in the SQL definition. You can also specify DEC ZONED.

DECIMAL EXTERNAL($length, scale$)
Specifies a string of characters that represent a number. The format is that of an SQL numeric constant.

- $length$: Overall length of the input field, in bytes. If you do not specify $length$, the length of the input field is determined from the POSITION specification. If you do not specify $length$ or POSITION, LOAD uses the default length for DECIMAL EXTERNAL, which is determined by using decimal precision.

- $scale$: Specifies the number of digits to the right of the decimal point. $scale$ must be an integer greater than or equal to 0, and it can be greater than $length$. The default value is 0.

If $scale$ is greater than $length$, or if the number of provided digits is less than the specified $scale$, the input number is padded on the left with zeros until the decimal point position is reached. If $scale$ is greater than the target $scale$, the source scale locates the implied decimal position. All fractional digits greater than the target scale are truncated. If $scale$ is specified and the target column has a data type of small integer or integer, the decimal portion of the input number is ignored. If a decimal point is present, its position overrides the field specification of $scale$.

FLOAT($length$)
Specifies either a 64-bit floating-point number or a 32-bit floating-point number. If $length$ is between 1 and 21 inclusive, the number is 32 bits in the s390 (HFP) format:

- **Bit 0**: Represents a sign (0 for plus and 1 for minus)
Bits 1-7  
Represent an exponent

Bits 8-31  
Represent a mantissa

*If* length* is between 1 and 24 inclusive, the number is 32 bits in the IEEE (BFP) format:

**Bit 0**  
Represents a sign (0 for plus and 1 for minus)

**Bits 1-8**  
Represent an exponent

**Bits 9-31**  
Represent a mantissa

*If* length* is not specified, or is between 22 and 53 inclusive, the number is 64 bits in the s390 (HFP) format:

**Bit 0**  
Represents a sign (0 for plus and 1 for minus)

**Bits 1-7**  
Represent an exponent

**Bits 8-63**  
Represent a mantissa.

*If* length* is not specified, or is between 25 and 53 inclusive, the number is 64 bits in the IEEE (BFP) format:

**Bit 0**  
Represents a sign (0 for “plus”, and 1 for “minus”)

**Bits 1-11**  
Represent an exponent

**Bits 12-63**  
Represent a mantissa.

You can also specify REAL for single-precision floating-point numbers and DOUBLE PRECISION for double-precision floating-point numbers.

**FLOAT EXTERNAL**(length)**

Specifies a string of characters that represent a number. The format is that of an SQL floating-point constant.

A specification of FLOAT(IEEE) or FLOAT(S390) does not apply for this format (string of characters) of floating-point numbers.

If you do not specify *length*, the length of the string is determined from the POSITION specification. If you do not specify *length* or POSITION, LOAD uses the default length for FLOAT, which is 4 bytes for single precision and 8 bytes for double precision.

**DATE EXTERNAL**(length)**

Specifies a character string representation of a date. You can specify a length or a specific date format.

The length, if unspecified, is the specified length on the LOCAL DATA LENGTH installation option, or, if none was provided, the default is 10 bytes. If you specify a length, it must be within the range of 8 - 254 bytes.
If a date format is not specified, dates can be in any of the following formats. You can omit leading zeros for month and day. You can include trailing blanks, but no leading blanks are allowed.

- *dd.mm/yyyy*
- *mm/dd/yyyy*
- *yyyy-mm-dd*
- Any local format that was defined when Db2 was installed

*(date-format)*

The format of the date representation, as shown in the following table.

**Table 31. Values for date-format**

<table>
<thead>
<tr>
<th>date-format</th>
<th>Format</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE_A</td>
<td><em>mm-dd-yyyy</em></td>
<td>10 bytes</td>
</tr>
<tr>
<td>DATE_B</td>
<td><em>mm-dd-yyyy</em></td>
<td>8 bytes</td>
</tr>
<tr>
<td>DATE_C</td>
<td><em>yyyy-mm-dd</em></td>
<td>10 bytes</td>
</tr>
<tr>
<td>DATE_D</td>
<td><em>yy-mm-dd</em></td>
<td>8 bytes</td>
</tr>
<tr>
<td>DATE_E</td>
<td><em>dd-mm-yyyy</em></td>
<td>10 bytes</td>
</tr>
<tr>
<td>DATE_F</td>
<td><em>dd-mm-yyyy</em></td>
<td>8 bytes</td>
</tr>
<tr>
<td>DATE_G</td>
<td><em>yyyy-ddd</em></td>
<td>8 bytes</td>
</tr>
<tr>
<td>DATE_H</td>
<td><em>yy-ddd</em></td>
<td>6 bytes</td>
</tr>
<tr>
<td>DATE_I</td>
<td><em>mmddyyyy</em></td>
<td>8 bytes</td>
</tr>
<tr>
<td>DATE_J</td>
<td><em>mmddyy</em></td>
<td>6 bytes</td>
</tr>
<tr>
<td>DATE_K</td>
<td><em>yyyyymmdd</em></td>
<td>8 bytes</td>
</tr>
<tr>
<td>DATE_L</td>
<td><em>yyyymmdd</em></td>
<td>6 bytes</td>
</tr>
<tr>
<td>DATE_M</td>
<td><em>ddmmyyyy</em></td>
<td>8 bytes</td>
</tr>
<tr>
<td>DATE_N</td>
<td><em>ddmmyy</em></td>
<td>6 bytes</td>
</tr>
<tr>
<td>DATE_O</td>
<td><em>yyyyyddd</em></td>
<td>7 bytes</td>
</tr>
<tr>
<td>DATE_P</td>
<td><em>yyddd</em></td>
<td>5 bytes</td>
</tr>
</tbody>
</table>

If a date format is specified for a field that is used in a field specification, the field specification must also use the specified date format.

**Notes:**

1. If the format includes separators, the separator can be any single-byte character that can be converted to a single-byte EBCDIC character. However, if the input file has a delimited format, you cannot specify the same character that is used for a delimiter, including the COLDEL, CHARDEL, and DECPT delimiters. For more information about delimiter restrictions, see "Loading delimited files" on page 325.

2. If a two-digit year is specified, it is expanded to a four-digit year. If the two-digit year is less than the sum of the two-digit current year plus 50, then the current century is used in the four-digit year. For example, assume that the current year is 2017. If the two-digit year is 67, 1967 is used. However, if the two-digit year is 66, 2066 is used.

**TIME EXTERNAL**

Specifies a character string representation of a time. You can specify a length or a specific time format.
The length, if unspecified, is the specified length on the LOCAL TIME LENGTH installation option, or, if none was provided, the default is 8 bytes. If you specify a length, it must be within the range of 4 - 254 bytes.

If a format is not specified, times can be in any of the following formats:
- \textit{hh:mm:ss}
- \textit{hh:mm} AM
- \textit{hh:mm} PM
- \textit{hh:mm:ss}
- Any local format that was defined when Db2 was installed

You can omit the \textit{mm} portion of the \textit{hh:mm} AM and \textit{hh:mm} PM formats if \textit{mm} is equal to 00. For example, 5 PM is a valid time, and can be used instead of 5:00 PM.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|l|}
\hline
\textit{time-format} value & Format & Length \\
\hline
\textbf{TIME\_A} & \textit{hh:mm:ss} & 8 bytes \\
\hline
\textbf{TIME\_B} & \textit{hh:mm} & 5 bytes \\
\hline
\textbf{TIME\_C} & \textit{hh:mm} AM or \textit{hh:mm} PM & 8 bytes \\
\hline
\textbf{TIME\_D} & \textit{hhmmss} & 6 bytes \\
\hline
\textbf{TIME\_E} & \textit{hhmm} & 4 bytes \\
\hline
\end{tabular}
\caption{Values for \textit{time-format}}
\end{table}

If a time format is specified for a field that is used in a field specification, the field specification must also use the specified time format.

\textbf{Note:}
1. If the format includes separators, the separator can be any single-byte character that can be converted to a single-byte EBCDIC character.

\textbf{TIMESTAMP EXTERNAL}

Specifies a character string representation of a time. You can specify a length or a specific timestamp format.

The default for \textit{length} is 26 bytes. If you specify a length, it must be within the range of 19 - 32 bytes.

If a timestamp format is not specified, timestamps can be in any of the following formats. \textit{nnnnnn} represents the number of microseconds, and can be 0 - 12 digits. You can omit leading zeros from the month, day, or hour parts of the timestamp; you can omit trailing zeros from the microseconds part of the timestamp.
- \textit{yyyy-mm-dd-hh:mm:ss}
- \textit{yyyy-mm-dd-hh:mm:ss.nnnnnn}
- \textit{yyyy-mm-dd hh:mm:ss.nnnnnn}

\textbf{timestamp-format}

The format of the timestamp representation, as shown in the following table.
Table 33. Values for timestamp-format

<table>
<thead>
<tr>
<th>timestamp-format</th>
<th>value</th>
<th>Format</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMESTAMP_A</td>
<td>yyyy-mm-dd-hh.mm.ss</td>
<td>19 bytes</td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP_B</td>
<td>yyyy-mm-dd-hh.mm.ss.nnnnn</td>
<td>26 bytes</td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP_C</td>
<td>yyyy:mmddhhmms</td>
<td>14 bytes</td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP_D</td>
<td>yymmddhhmmss</td>
<td>12 bytes</td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP_E</td>
<td>yyyy:mmddhhmmssnnnnnn</td>
<td>20 bytes</td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP_F</td>
<td>yymmddhhmmssnnnnnnn</td>
<td>18 bytes</td>
<td></td>
</tr>
</tbody>
</table>

If a timestamp format is specified for a field that is used in a field specification, the field specification must also use the specified timestamp format.

Notes:
1. If the format includes separators, the separator can be any single-byte character that can be converted to a single-byte EBCDIC character.
2. If a two-digit year is specified, it is expanded to a four-digit year. If the two-digit year is less than the sum of the two-digit current year plus 50, then the current century is used in the four-digit year. For example, assume that the current year is 2017. If the two-digit year is 67, 1967 is used. However, if the two-digit year is 66, 2066 is used.

**TIMESTAMP WITH TIME ZONE EXTERNAL**(length)

Specifies a character string representation of a timestamp with time zone. The default for `length` is 33 bytes. If you specify a length, it must be within the range of 26 to 39 bytes.

Timestamp with time zone can be in any of the following formats. `nnnnnn` represents the number of digits in the fractional seconds, and can be 0 - 12 digits. You can omit leading zeros from the month, day, or hour parts of the timestamp; you can omit trailing zeros from the fractional seconds part of the timestamp.

- `yyyy-mm-dd-hh.mm.ss.nnnnn±th:tm`
- `yyyy-mm-dd-hh.mm.ss.nnnnn ±th:tm`
- `yyyy-mm-dd hh:mm:ss.nnnnn±th:tm`
- `yyyy-mm-dd hh:mm:ss.nnnnn ±th:tm`

**ROWID**

Specifies a row ID. The input data must be a valid value for a row ID; Db2 does not perform any conversions.

A field specification for a row ID column is not allowed if the row ID column was created with the GENERATED ALWAYS option.

If the row ID column is part of the partitioning key, LOAD INTO TABLE PART is not allowed; specify LOAD INTO TABLE instead.

**BLOB**

Specifies a BLOB field. You must specify the length in bytes in a 4-byte binary field that precedes the data. (The length does not include the 4-byte field itself.) The length field must start in the column that is specified as `start` in the POSITION option. If `:end` is used, it is ignored.

**CLOB**

Specifies a CLOB field. You must specify the length in bytes in a 4-byte binary
field that precedes the data. (The length does not include the 4-byte field itself.) The length field must start in the column that is specified as \textit{start} in the \textsc{position} option. If \textit{:end} is used, it is ignored.

**MIXED**

Specifies that the input field contains mixed SBCS and DBCS data. If MIXED is specified, any required CCSID conversions use the mixed CCSID for the input data; if MIXED is not specified, any such conversions use the SBCS CCSID for the input data.

**DBCLOB**

Specifies a DBCLOB field. You must specify the length in double-byte characters in a 4-byte binary field that precedes the data. (The length does not include the 4-byte field itself.) The length field must start in the column that is specified as \textit{start} in the \textsc{position} option. If \textit{:end} is used, it is ignored.

**DECFLOAT** \texttt{(length)}

Specifies either a 128-bit decimal floating-point number or a 64-bit decimal floating-point number. The value of the length must be either 16 or 34. If the length is 16, the number is in 64 bit decimal floating-point number format. If the length is 34, the number is in 128 bit decimal floating-point format. If the length is not specified, the number is in 128 bit decimal floating-point format.

**DECFLOAT EXTERNAL** \texttt{(length)}

Specifies a string of characters that represent a number. The format is an SQL numeric constant. If you do not specify a length, the length of the string is determined from the \textsc{position} specification. If you do not specify a length or \textsc{position}, \textsc{load} uses the default length for \textsc{decfloat}.

**XML**

Specifies the input field type is XML. Field type XML can only be loaded to a XML column. Specify XML when loading the XML value directly from the input record. If the format of the input record is in nondelimited, you must specify a 2 byte length field precedes the actual data value.

\textsc{binaryxml} Specifies that the XML document to be loaded using the file reference variables is in binary XML format.

**PRESERVE WHITESPACE**

Specifies that the white space in the XML column is preserved. The default is not to preserve the white space.

**DEFAULTIF** \texttt{field-selection-criterion}

Describes a condition that causes the Db2 column to be loaded with its default value. You can write the \texttt{field-selection-criterion} with the same options as described under \texttt{field-selection-criterion}. If the contents of the DEFAULTIF field match the provided character constant, the field that is specified in \texttt{field-specification} is loaded with its default value.

If the DEFAULTIF field is defined by the name of a VARCHAR or VARGRAPHIC field, Db2 takes the length of the field from the 2-byte binary field that appears before the data portion of the VARCHAR or VARGRAPHIC field.

Character-string constants should be specified in LOAD utility control statements in the character set that matches the input data record. Specify EBCDIC constants in the LOAD control statement if your data is in EBCDIC and specify UNICODE constants if your data is in UNICODE. You may also code the DEFAULTIF condition using the hexadecimal form. For example, if the input data is in EBCDIC and the control statement is in UTF-8, use \texttt{(1:1)=X'31'} in the condition rather than \texttt{(1:1)='1'}.  

Chapter 19. LOAD  303
You can use the DEFAULTIF attribute with the ROWID keyword. If the condition is met, the column is loaded with a value that Db2 generates.

You cannot specify the DEFAULTIF option for XML columns.

**NULLIF field-selection-criterion**

Describes a condition that causes the Db2 column to be loaded with NULL. You can write the field-selection-criterion with the same options as described under field-selection-criterion. If the contents of the NULLIF field match the provided character constant, the field that is specified in field-specification is loaded with NULL.

If the NULLIF field is defined by the name of a VARCHAR or VARGRAPHIC field, Db2 takes the length of the field from the 2-byte binary field that appears before the data portion of the VARCHAR or VARGRAPHIC field.

To load a null value into a BLOBF, CLOBF, or DBCLOBF field, use a null input file name.

Character-string constants should be specified in LOAD utility control statements in the character set that matches the input data record. Specify EBCDIC constants in the LOAD control statement if your data is in EBCDIC and specify UNICODE constants if your data is in UNICODE. You may also code the NULLIF condition using the hexadecimal form. For example, if the input data is in EBCDIC and the control statement is in UTF-8, use (1:1)=X’31’ in the condition rather than (1:1)=’1’.

The fact that a field in the output table is loaded with NULL does not change the format or function of the corresponding field in the input record. The input field can still be used in a field selection criterion. For example, assume that a LOAD statement has the following field specification:

```sql
(FIELD1 POSITION(*) CHAR(4)
 FIELD2 POSITION(*) CHAR(3) NULLIF(FIELD1='SKIP')
 FIELD3 POSITION(*) CHAR(5))
```

Assume also that LOAD is to process the following source record:

```
SKIP FLD03
```

In this example, the record is loaded as follows:

**FIELD1**

Has the value 'SKIP'.

**FIELD2**

Is NULL (not ' ' as in the source record).

**FIELD3**

Has the value 'FLD03'.

You cannot use the NULLIF parameter with the ROWID keyword because row ID columns cannot be null.

**Field selection criterion**

Describes a condition that causes the Db2 column to be loaded with NULL or with its default value.

**Related concepts:**

- Constants (Db2 SQL)

**Related tasks:**

- “Preparing Db2 internal format input records that are not generated by UNLOAD for LOAD” on page 314
Related reference:
- STRIP (Db2 SQL)
- EDITPROCs and VALIDPROCs for handling basic and reordered row formats (Db2 Administration Guide)
- Chapter 34, “TEMPLATE,” on page 793
- Db2 Sort for z/OS

Related information:
- Converting basic row format table spaces with edit and validation routines to reordered row format (Db2 Administration Guide)
- DFSORT Application Programming Guide

Before running LOAD

Certain activities might be required before you run the LOAD utility, depending on your situation.

You cannot run the LOAD utility on the DSNB01 or DSNB06 databases, except to add rows to the following catalog tables:
- SYSSTRINGS
- MODESELECT
- LUMODES
- LULIST
- USERNAMES
- LUNAMES
- LOCATIONS
- IPNAMES

If you are using LOAD for a partition-by-growth table space, you can load data only at the table space level, not at the partition level.

Preprocessing input data

No sorting of the data rows occurs during LOAD processing. Rows are loaded in the physical sequence in which they are found.

**Recommendation:** Sort your input records in clustering sequence before loading the data.

You should also:
- Ensure that no duplicate keys exist for unique indexes.
- Correct check constraint violations and referential constraint violations in the input data set.
- Ensure that any input data that is provided for a security label column is a valid security label. Security label columns are defined with the AS SECURITY LABEL clause. These columns are used for multilevel security with row-level granularity.

When loading data into a segmented table space, sort your data by table to ensure that the data is loaded in the best physical organization.
Loading data by using a cursor

Before you can load data by using a cursor, you need to bind the DSNUT111 package at each location from which you plan to load data. A local package for DSNUT111 is bound by installation job DSNTIJSG when you install or migrate to a new version of Db2 for z/OS.

The following example statement binds the DSNUT111 package at a remote location:

```
BIND PACKAGE(location.DSNUT111)
    MEMBER(DSNUGSQL) -
    ACTION(ADD) ISOLATION(CS) ENCODING(EBCDIC) -
    VALIDATE(BIND) CURRENTDATA(NO) -
    LIBRARY('prefix.SDSNDBRM')
```

You can improve the performance of cross-loading from a remote Db2 for z/OS Db2 11 subsystem in new-function mode to a local Db2 for z/OS Db2 11 subsystem in new-function mode by binding the DSNUTIL and DSNUT111 packages again on the local and remote subsystems with the DBPROTOCOL(DRDACBF) option after you have converted to new-function mode.

**Important:** Bind the DSNUTIL package with the DBPROTOCOL(DRDACBF) option only if you do not plan to use the EXEC SQL utility control statement to perform remote updates.

The following example statements bind the DSNUTIL and DSNUT111 packages on the local subsystem, and bind the DSNUT111 package on the remote subsystem:

```
BIND PACKAGE(DSNUTIL) MEMBER(DSNUGSQL) -
    ACTION(ADD) ISOLATION(CS) ENCODING(EBCDIC) -
    VALIDATE(BIND) CURRENTDATA(NO) -
    DBPROTOCOL(DRDACBF) -
    LIBRARY('prefix.SDSNDBRM')

BIND PACKAGE(DSNUT111) MEMBER(DSNUGSQL) -
    ACTION(ADD) ISOLATION(CS) ENCODING(EBCDIC) -
    VALIDATE(BIND) CURRENTDATA(NO) -
    DBPROTOCOL(DRDACBF) -
    LIBRARY('prefix.SDSNDBRM')

BIND PACKAGE(location.DSNUT111) MEMBER(DSNUGSQL) -
    ACTION(ADD) ISOLATION(CS) ENCODING(EBCDIC) -
    VALIDATE(BIND) CURRENTDATA(NO) -
    DBPROTOCOL(DRDACBF) -
    LIBRARY('prefix.SDSNDBRM')
```

Running LOAD on a table with a spatial index

You cannot run the LOAD utility to load data into a table on which a spatial index is defined. You need to drop the spatial index, run LOAD on the table, and then create the spatial index again.

**Related concepts:**

- [Multilevel security (Managing Security)](multilevel_security_managing_security)
**Data sets that LOAD uses**

The LOAD utility requires certain data sets depending on the options that you specify. Include statements in your JCL for each data set that is needed. Alternatively, for some of these data sets, you can use the TEMPLATE utility to dynamically allocate them.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Default DD name</th>
<th>LOAD option to specify a different DD name or template name</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-stream</td>
<td>SYSIN</td>
<td>None</td>
<td>An input data stream that contains the utility control statement.</td>
<td>Yes</td>
</tr>
<tr>
<td>data set</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output data</td>
<td>SYSPRINT</td>
<td>None</td>
<td>An output data set for utility messages.</td>
<td>Yes</td>
</tr>
<tr>
<td>set</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input data set</td>
<td>SYSREC</td>
<td>INDDN</td>
<td>An input data set that contains the data that is to be loaded.</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error data set</td>
<td>SYSERR</td>
<td>ERRDDN</td>
<td>A work data set for error processing.</td>
<td>Yes</td>
</tr>
<tr>
<td>Discard data</td>
<td>SYSDISC</td>
<td>DISCARDDN</td>
<td>A work data set that contains copies of records that are not loaded.</td>
<td>Yes</td>
</tr>
<tr>
<td>set</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sort message</td>
<td>UTPRINT</td>
<td>None</td>
<td>An output data set that contains messages from the sort program. These</td>
<td>Yes</td>
</tr>
<tr>
<td>data set</td>
<td></td>
<td></td>
<td>messages are usually routed to SYSOUT or DUMMY.</td>
<td></td>
</tr>
<tr>
<td>Copy data</td>
<td>SYSCOPY</td>
<td>COPYDDN and RECOVERYDDN</td>
<td>One to four output data sets that contain image copy data sets.</td>
<td>Required for inline copies.</td>
</tr>
<tr>
<td>sets</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data set</td>
<td>Default DD name</td>
<td>LOAD option to specify a different DD name or template name</td>
<td>Description</td>
<td>Required?</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FlashCopy image copies</td>
<td>None</td>
<td>FCCOPYDDN</td>
<td>A VSAM data set for the output FlashCopy image copy of each partition or piece.</td>
<td>Required for inline FlashCopy image copies (FLASHCOPY YES or FLASHCOPY CONSISTENT is specified).</td>
</tr>
<tr>
<td>Sort message data set</td>
<td>RNPRINnn</td>
<td>None</td>
<td>An output data set that contains messages from the sort program. These messages are usually routed to SYSOUT or DUMMY.</td>
<td>Required when collecting distribution statistics for column groups (COLGROUP and FREQVAL are specified).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$nn$ is a number from 01 to the number of parallel subtasks.</td>
</tr>
<tr>
<td>Sort message data set</td>
<td>STPRIN01</td>
<td>None</td>
<td>An output data set that contains messages from the sort program. These messages are usually routed to SYSOUT or DUMMY. LOAD dynamically allocates the STPRIN01 data set if UTPRINT is allocated to SYSOUT.</td>
<td>Required when collecting distribution statistics for column groups (COLGROUP and FREQVAL are specified) or when statistics are being collected on at least one data-partitioned secondary index.</td>
</tr>
<tr>
<td>Mapping data set</td>
<td>SYSMAP</td>
<td>MAPDDN</td>
<td>A work data set for mapping the identifier of a table row to the input record that caused an error.</td>
<td>Required for the following situations:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• If referential constraints exist and ENFORCE(CONSTRAINTS) is specified. (This option is the default.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• For discard processing when loading one or more tables that have unique indexes.</td>
</tr>
<tr>
<td>Sort input data set</td>
<td>SYSUT1</td>
<td>WORKDDN</td>
<td>A temporary work data set for sort input. If the input data set is on tape, and you use a templates for the SYSUT1 data set, include the SPACE parameter in the TEMPLATE utility control statements.</td>
<td>Required if referential constraints exist and ENFORCE(CONSTRAINTS) is specified (This option is the default).</td>
</tr>
<tr>
<td>Sort output data set</td>
<td>SORTOUT</td>
<td>WORKDDN</td>
<td>A temporary work data set for sort output.</td>
<td>Required if referential constraints exist and ENFORCE(CONSTRAINTS) is specified (This option is the default).</td>
</tr>
</tbody>
</table>
### Table 34. Data sets that LOAD uses (continued)

<table>
<thead>
<tr>
<th>Data set</th>
<th>Default DD name</th>
<th>LOAD option to specify a different DD name or template name</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sort work data sets</td>
<td>SW\textit{nn}WK\textit{mm} or SORTWK\textit{mm}</td>
<td>None</td>
<td>Temporary data sets for sort input and output when sorting keys. If index build parallelism is used, the DD names have the form SW\textit{nn}WK\textit{mm}. If index build parallelism is not used, the DD names have the form SORTWK\textit{mm}.</td>
<td>Required if any indexes are to be built or if a sort is required for processing errors.</td>
</tr>
<tr>
<td>Sort work data sets</td>
<td>ST01WK\textit{mm}</td>
<td>None</td>
<td>Temporary data sets for sort input and output when collecting inline statistics on at least one data-partitioned secondary index or when the COLGROUP option is specified.</td>
<td>Required when collecting inline statistics on at least one data-partitioned secondary index.</td>
</tr>
<tr>
<td>Sort work data sets</td>
<td>RN\textit{mm}WK\textit{mm}</td>
<td>None</td>
<td>Temporary data sets for sort input and output when collecting distribution statistics for column groups. The DD names have the form RN\textit{mm}WK\textit{mm}, where \textit{mm} is the subtask number, and \textit{ww} is a sequence number for the data set that is allocated for that subtask.</td>
<td>Required when collecting inline statistics on at least one data-partitioned secondary index.</td>
</tr>
<tr>
<td>Sort work data sets</td>
<td>SORTWK01</td>
<td>None</td>
<td>Temporary data sets for sort input and output when collecting frequency statistics.</td>
<td>Required when collecting frequency statistics.</td>
</tr>
</tbody>
</table>

**Note:**

1. If the DYNALLOC parm of the sort program is not turned on, you need to allocate the data set. Otherwise, the sort program dynamically allocates the temporary data set.

2. **Recommendation:** Use dynamic allocation by specifying the SORTDEVT option in the LOAD statement so that you do not have to write and maintain these DD statements.

---

### Defining work data sets

Use the formulas and instructions in The following table to calculate the size of work data sets for LOAD. Each row in the table lists the DD name that is used to identify the data set and either formulas or instructions that you should use to determine the size of the data set. The key for the formulas is located at the bottom of the table.

#### Table 35. Size of work data sets for LOAD jobs

<table>
<thead>
<tr>
<th>Work data set</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SORTOUT</td>
<td>max(f,e)</td>
</tr>
</tbody>
</table>
### Table 35. Size of work data sets for LOAD jobs (continued)

<table>
<thead>
<tr>
<th>Work data set</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST01WKnn</td>
<td>$2 \times (\text{maximum record length} \times \text{numcols} \times (\text{count} + 2) \times \text{number of indexes})$</td>
</tr>
<tr>
<td>SYSDISC</td>
<td>Same size as input data set</td>
</tr>
<tr>
<td>SYSERR</td>
<td>$e$</td>
</tr>
</tbody>
</table>
| SYSMAP        | - Simple table space for discard processing: $m$
|               | - Partitioned or segmented table space without discard processing: $\max(m,e)$ |
| SYSUT1        | - Simple table space: $\max(k,e)$
|               | - Partitioned or segmented table space: $\max(k,e,m)$
|               | If you specify an estimate of the number of keys with the SORTKEYS option:
|               | $\max(f,e)$ for a simple table space
|               | $\max(f,e,m)$ for a partitioned or segmented table space |

**Note:**

<table>
<thead>
<tr>
<th>variable</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k$</td>
<td>Key calculation</td>
</tr>
<tr>
<td>$f$</td>
<td>Foreign key calculation</td>
</tr>
<tr>
<td>$m$</td>
<td>Map calculation</td>
</tr>
<tr>
<td>$e$</td>
<td>Error calculation</td>
</tr>
<tr>
<td>$\max()$</td>
<td>Maximum value of the specified calculations</td>
</tr>
<tr>
<td>numcols</td>
<td>Number of key columns to concatenate when you collect frequent values from the specified index</td>
</tr>
<tr>
<td>count</td>
<td>Number of frequent values that Db2 is to collect</td>
</tr>
<tr>
<td>maximum record length</td>
<td>Maximum record length of the SYSCOLDISTSTATS record that is processed when collecting frequency statistics (You can obtain this value from the RECLENGTH column in SYSTABLES.)</td>
</tr>
</tbody>
</table>

**Calculating the key: $k$**

If a mix of data-partitioned secondary indexes and nonpartitioned indexes exists on the table that is being loaded or a foreign key exists that is exactly indexed by a data-partitioned secondary index, use this formula: $\max(\text{longest index key} + 15, \text{longest foreign key} + 15) \times (\text{number of extracted keys})$.

Otherwise, use this formula: $\max(\text{longest index key} + 13, \text{longest foreign key} + 13) \times (\text{number of extracted keys})$.

For nonpadded indexes, the length of the longest key means the maximum possible length of a key with all varying-length columns padded to their maximum lengths, plus 2 bytes for each varying-length column.

**Calculating the number of extracted keys:**

1. Count 1 for each index.
2. Count 1 for each foreign key that is not exactly indexed (that is, where foreign key and index definitions do not correspond identically).
3. For each foreign key that is exactly indexed (that is, where foreign key and index definitions correspond identically):
   a. Count 0 for the first relationship in which the foreign key participates if the index is not a data-partitioned secondary index. Count 1 if the index is a data-partitioned secondary index.
   b. Count 1 for subsequent relationships in which the foreign key participates (if any).
4. Multiply count by the number of rows that are to be loaded.

• Calculating the foreign key: \( f \)
   If a mix of data-partitioned secondary indexes and nonpartitioned indexes exists on the table that is being loaded or a foreign key exists that is exactly indexed by a data-partitioned secondary index, use this formula: 
   \[ \text{max(longest foreign key} + 15) \times \text{number of extracted keys} \]
   Otherwise, use this formula: 
   \[ \text{max(longest foreign key} + 13) \times \text{number of extracted keys} \]

• Calculating the map: \( m \)
   The data set must be large enough to accommodate one map entry (length = 21 bytes) per table row that is produced by the LOAD job.

• Calculating the error: \( e \)
   The data set must be large enough to accommodate one error entry (length = 560 bytes) per defect that is detected by LOAD (for example, conversion errors, unique index violations, violations of referential constraints).

• Calculating the number of possible defects:
   – For discard processing, if the discard limit is specified, the number of possible defects is equal to the discard limit.
     
     If the discard limit is the maximum, calculate the number of possible defects by using the following formula:
     
     \[ \text{number of input records} + \]
     \[ (\text{number of unique indexes} \times \text{number of extracted keys}) + \]
     \[ (\text{number of relationships} \times \text{number of extracted foreign keys}) \]
   – For nondiscard processing, the data set is not required.

Allocating twice the space that is used by the input data sets is usually adequate for the sort work data sets. Two or three large SORTWK\( \text{nn} \) data sets are preferable to several small ones.

Sort work data sets cannot span volumes. Smaller volumes require more sort work data sets to sort the same amount of data; therefore, large volume sizes can reduce the number of needed sort work data sets. It is recommended that at least 1.2 times the amount of data to be sorted be provided in sort work data sets on disk.

Tape devices are not supported for sort work data sets.

Related concepts:
“Data sets that online utilities use” on page 16

Related reference:

Db2 Sort for z/OS

“Syntax and options of the TEMPLATE control statement” on page 793

Related information:

DFSORT Application Programming Guide
Concurrency and compatibility for LOAD

Depending on the specified LOAD utility options and the target object, certain processes and utilities can run at the same time as LOAD.

Claims and drains

The following table shows the claim classes that LOAD drains and the restrictive states the utility sets.

Table 36. Claim classes of LOAD operations

<table>
<thead>
<tr>
<th>Target</th>
<th>LOAD SHRLEVEL NONE</th>
<th>LOAD PART SHRLEVEL NONE</th>
<th>LOAD SHRLEVEL CHANGE</th>
<th>LOAD PART SHRLEVEL CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table space, index, or physical partition of a table space or index space</td>
<td>DA/UTUT</td>
<td>DA/UTUT</td>
<td>CW/UTRW</td>
<td>CW/UTRW</td>
</tr>
<tr>
<td>Nonpartitioned secondary index¹</td>
<td>DA/UTUT</td>
<td>DR</td>
<td>CW/UTRW</td>
<td>CW/UTRW</td>
</tr>
<tr>
<td>Data-partitioned secondary index²</td>
<td>DA/UTUT</td>
<td>DA/UTUT</td>
<td>CW/UTRW</td>
<td>CW/UTRW</td>
</tr>
<tr>
<td>Index logical partition³</td>
<td>None</td>
<td>DA/UTUT</td>
<td>None</td>
<td>CW/UTRW</td>
</tr>
<tr>
<td>Primary index (with ENFORCE option only)</td>
<td>DW/UTRO</td>
<td>DW/UTRO</td>
<td>CR/UTR</td>
<td>CR/UTR</td>
</tr>
<tr>
<td>RI dependents</td>
<td>CHKP (NO)</td>
<td>CHKP (NO)</td>
<td>CHKP (NO)</td>
<td>CHKP (NO)</td>
</tr>
</tbody>
</table>

Legend:
- CHKP (NO): Concurrently running applications do not see CHECK-pending status after commit.
- CR: Claim the read claim class.
- CW: Claim the write claim class.
- DA: Drain all claim classes, no concurrent SQL access.
- DR: Drain the repeatable read class, no concurrent access for SQL repeatable readers.
- DW: Drain the write claim class, concurrent access for SQL readers.
- UTUT: Utility restrictive state, exclusive control.
- UTRO: Utility restrictive state, read-only access allowed.
- UTR: Utility restrictive state, read/write access allowed.
- None: Object is not affected by this utility.

Note:
1. Includes the document ID indexes and node ID indexes over non-partitioned XML table spaces and XML indexes.
2. Includes document ID indexes and node ID indexes over partitioned XML table spaces.
3. Includes logical partitions of an XML index over partitioned table spaces.
4. For nonpartitioned secondary indexes, LOAD PART drains only the logical partition.

Compatibility

Db2 treats individual data and index partitions as distinct target objects. Utilities that operate on different partitions of the same table space or index space are compatible.

The following table shows whether other utilities are compatible with LOAD and can run concurrently on the same target object. The target object can be a table space, an index space, or a partition of a table space or index space.
<table>
<thead>
<tr>
<th>Action</th>
<th>LOAD SHRLEVEL NONE</th>
<th>LOAD SHRLEVEL CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACKUP SYSTEM</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CHECK DATA DELETE NO</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CHECK DATA DELETE YES</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CHECK INDEX</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CHECK LOB</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>COPY INDEXSPACE SHRLEVEL CHANGE</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>COPY INDEXSPACE SHRLEVEL REFERENCE</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>COPY TABLESPACE SHRLEVEL CHANGE</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>COPY TABLESPACE SHRLEVEL REFERENCE</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>COPYTOCOPY</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>DIAGNOSE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LOAD SHRLEVEL CHANGE</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>LOAD SHRLEVEL NONE</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MERGECOPY</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>MODIFY RECOVERY</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>MODIFY STATISTICS</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>QUIESCE</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REBUILD INDEX</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RECOVER (no options)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RECOVER ERROR RANGE</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RECOVER TOCOPY or TORBA</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REORG INDEX</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REORG TABLESPACE UNLOAD CONTINUE or PAUSE</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REORG TABLESPACE UNLOAD ONLY or EXTERNAL</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REPAIR DUMP or VERIFY</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REPAIR LOCATE KEY or RID DELETE or REPLACE</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REPAIR LOCATE TABLESPACE PAGE REPLACE</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REPORT</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>RESTORE SYSTEM</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RUNSTATS INDEX SHRLEVEL CHANGE</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>RUNSTATS INDEX SHRLEVEL REFERENCE</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Table 37. Compatibility of LOAD with other utilities (continued)

<table>
<thead>
<tr>
<th>Action</th>
<th>LOAD SHRLEVEL NONE</th>
<th>LOAD SHRLEVEL CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUNSTATS TABLESPACE SHRLEVEL CHANGE</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>RUNSTATS TABLESPACE SHRLEVEL REFERENCE</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>STOSPACE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UNLOAD</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

SQL operations and other online utilities on the same target partition are incompatible.

Related concepts:

Claims and drains (Db2 Performance)

Preparing Db2 internal format input records that are not generated by UNLOAD for LOAD

Before you load records that are in Db2 internal format, but are not generated by the UNLOAD utility, you need to modify the input records.

About this task

For LOAD input records that are in Db2 internal format, the seventh bit of the first byte in each record needs to indicate whether the record is in basic row format or reordered row format. A value of 0 in the seventh bit of the first byte indicates that the record is in basic row format. A value of 1 indicates that the record is in reordered row format.

Procedure

If your input records are in Db2 internal format, and you did not use UNLOAD with FORMAT INTERNAL to generate the input data, you need to set the seventh bit of the first byte of each record to indicate the row format of the record. To avoid extra row format conversions that degrade performance, ensure that input data rows are in the row format of the target table space or partition.

Related reference:

“Syntax and options of the LOAD control statement” on page 242

When to use SORTKEYS NO

The SORTKEYS value determines when you can restart a LOAD job on a table space that has LOB columns.

- The default value for SORTKEYS is SORTKEYS 0. If you plan to load a table that has LOB columns using LOAD RESUME YES SHRLEVEL NONE, and you might need to restart the LOAD job with RESTART(CURRENT), you must specify SORTKEYS NO.
- The point at which you can restart LOAD REPLACE SHRLEVEL NONE on a table that has no LOB columns depends on whether you specify SORTKEYS NO:
  - If you specify SORTKEYS NO, you can restart with RESTART(CURRENT).
Loading variable-length data

You can load variable-length data by using the LOAD utility.

Procedure

To load variable-length data:

Include a 2-byte binary length field before each field of variable-length data. The value in that field depends on the data type of the column into which you load the data. Use:

- The number of single-byte characters if the data type is VARCHAR
- The number of double-byte characters if the data type is VARGRAPHIC

For example, assume that you have a variable-length column that contains X'42C142C142C2', which might be interpreted as either six single-byte characters or three double-byte characters. With the two-byte length field, use:

- X'0006'X'42C142C142C2' to signify six single-byte characters in a VARCHAR column
- X'0003'X'42C142C142C2' to signify three double-byte characters in a VARGRAPHIC column

How LOAD orders loaded records

The LOAD utility loads records into a table space in the order in which they appear in the input stream. It does not sort the input stream, and it does not insert records in sequence with existing records, even if a clustering index exists.

To achieve clustering when loading an empty table or replacing data, sort the input stream. When adding data to a clustered table, consider reorganizing the table after running LOAD.

Because rows with duplicate key values for unique indexes fail to be loaded, any records that are dependent on such rows either:

- Fail to be loaded because they would cause referential integrity violations (if you specify ENFORCE CONSTRAINTS)
- Are loaded without regard to referential integrity violations (if you specify ENFORCE NO)

As a result, violations of referential integrity might occur. Such violations can be detected by LOAD (without the ENFORCE(NO) option) or by CHECK DATA.

Replacing data with LOAD

You can use the LOAD utility to replace data in a table space that has one or more tables.

Procedure

To replace data with LOAD:
Specify the REPLACE option in the LOAD utility control statement. This option specifies that all data in the table space is to be replaced. Alternatively, you can load new records into a table space without deleting the existing rows by using the RESUME option.

When you specify LOAD REPLACE, determine what other LOAD options to specify depending on the following implications:

**How data sets are processed**

Db2 processes data sets depending on the LOAD options that you specify. If you run LOAD REPLACE without the REUSE option, data sets that are not user-managed are deleted before the LOAD utility runs. The LOAD utility defines a new data set with a control interval that matches the page size.

**How row format is affected**

When you run LOAD REPLACE with the ROWFORMAT RRF option on a table space or partition that is in basic row format, LOAD converts the table space or partition to the reordered row format. If the ROWFORMAT BRF option is specified, existing basic row format table spaces are not converted to reordered row format. If the clause EDITPROC or VALIDPROC is used in a table space or partition, the table space or partition remains in basic format after the LOAD REPLACE. For table spaces that contain some partitions in basic row format and some partitions in reordered row format, LOAD REPLACE converts the partitions that are in basic row format to reordered row format.

**How logging is handled**

The LOAD REPLACE or PART REPLACE with LOG YES option logs only the reset and not each deleted row. To see what rows are being deleted, use the SQL DELETE statement.

Running LOAD REPLACE has the following effects on restrictive states:

**REORG-pending**

If an object is in REORG-pending status, you can run LOAD REPLACE on the entire table space, which resets REORG-pending status. You can also run LOAD PART REPLACE or RESUME on any partitions that are not in REORG-pending status. In this situation, no other LOAD operations are allowed.

**Advisory REORG-pending**

If an object is in advisory REORG-pending status, you can run LOAD REPLACE on the entire table space, which resets advisory REORG-pending status. The exception is pending limit key changes. LOAD REPLACE does not materialize those changes or reset advisory REORG-pending status. In that case, you must run the REORG TABLESPACE utility. Then, you can run LOAD REPLACE. (You can continue to use LOAD REPLACE to materialize immediate alter limit key changes, which are indicated by REORG-pending status instead of advisory REORG-pending status. Immediate alter limit key changes occur for a partitioned table space with index-controlled partitioning and any alter limit key operations that occur before Db2 11 new-function mode.)

**REBUILD-pending**

If an object is in REBUILD-pending status, you can run LOAD REPLACE on the entire table space, which resets REBUILD-pending status. You can also run LOAD PART REPLACE or RESUME on any partitions. If these partitions are in REBUILD-pending status, a LOAD PART REPLACE or RESUME resets that status.
Advisory REBUILD-pending
If an object is in advisory REBUILD-pending status, you can run LOAD REPLACE on the entire table space, which resets advisory REBUILD-pending status.

REFRESH-pending
If a user-defined table space is in REFRESH-pending (REFP) status, you can replace the data by using LOAD REPLACE.

Examples

Example of replacing one table in a single-table table space
The following control statement specifies that LOAD is to replace one table in a single-table table space.

```
LOAD DATA
REPLACE
INTO TABLE DSN8B10.DEPT
( DEPTNO POSITION (1) CHAR(3),
  DEPTNAME POSITION (5) VARCHAR,
  MGRNO POSITION (37) CHAR(6),
  ADMRDEPT POSITION (44) CHAR(3),
  LOCATION POSITION (48) CHAR(16) )
ENFORCE NO
```

Example of replacing one table in a multiple-table table space
LOAD works on an entire table space. Therefore, be careful when using LOAD REPLACE on a table space with multiple tables. To replace all rows in a multiple-table table space, you must work with one table at a time by using the RESUME YES option on all but the first table. For example, if you have two tables in a table space, take the following steps:

1. Use LOAD REPLACE on the first table as shown in the following control statement. This option removes data from the table space and replaces just the data for the first table.
   
   ```
   LOAD DATA CONTINUEIF(72:72)='X'
   REPLACE
   INTO DSN8B10.TOPTVAL
   ( MAJSYS POSITION (2) CHAR(1),
     ACTION POSITION (4) CHAR(1),
     OBJECT POSITION (6) CHAR(2),
     SRCHCRIT POSITION (9) CHAR(2),
     SCRTYPE POSITION (12) CHAR(1),
     HEADTXT POSITION (80) CHAR(50),
     SELTXT POSITION (159) CHAR(79),
     INFOTXT POSITION (238) CHAR(71),
     HELPTXT POSITION (317) CHAR(71),
     PFKTXT POSITION (396) CHAR(71),
     DSPINDEX POSITION (475) CHAR(2) )
   ```

2. Use LOAD with RESUME YES on the second table as shown in the control statement in the following example. This option adds the records for the second table without deleting the data in the first table.

   ```
   LOAD DATA CONTINUEIF(72:72)='X'
   RESUME YES
   INTO DSN8B10.TDSTXT
   ( DSPINDEX POSITION (2) CHAR(2),
     LINENO POSITION (6) CHAR(2),
     DSPLINE POSITION (80) CHAR(79) )
   ```

If you want to replace just one table in a multiple-table table space, delete all rows in the table, and then use LOAD with RESUME YES. For example,
assume that you want to replace all the data in DSN8B10.TDSPTXT without changing any data in DSN8B10.TOPTVAL. In this case, take the following steps:

1. Delete all the rows from DSN8B10.TDSPTXT by using the following SQL DELETE statement:

   ```sql
   EXEC SQL
   DELETE FROM DSN8B10.TDSPTXT
   ENDEXEC
   ```

   **Tip:** The mass delete works most quickly on a segmented table space.

2. Use the LOAD job that is shown in the following figure to replace the rows in that table.

   ```sql
   LOAD DATA CONTINUEIF(72:72)='X'
   RESUME YES
   INTO DSN8B10.TDSPTXT
   (DSPINDEX POSITION (2) CHAR(2),
    LINENO POSITION (6) CHAR(2),
    DSPLINE POSITION (80) CHAR(79) )
   ```

**Related concepts:**

- [Improved data availability when altering limit keys (Db2 for z/OS What's New?)](#)

**Related reference:**

- [“Syntax and options of the LOAD control statement” on page 242](#)
- Chapter 28, “REORG TABLESPACE,” on page 553
- Appendix B, “Advisory or restrictive states,” on page 1037

---

### Loading tables with special column types by using generated LOAD statements

When you run the UNLOAD utility or the REORG utility with the UNLOAD EXTERNAL or DISCARD option, Db2 generates a LOAD statement for the unloaded data. You can then use this LOAD statement to load the unloaded data into any table that has a compatible format.

**About this task**

However, because the following types of columns can contain generated values, they need special consideration:

- ROWID columns
- Identity columns
- Row change timestamp columns
- Row-begin and row-end columns
- Transaction-start-ID columns
- Generated expression columns

For these table column types, the generated LOAD statement contains dummy fields. The following table lists those dummy fields.

<table>
<thead>
<tr>
<th>Source table column type</th>
<th>Dummy fields in the generated LOAD statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROWID with GENERATED ALWAYS</td>
<td>DSN_ROWID</td>
</tr>
<tr>
<td>Identity column with GENERATED ALWAYS</td>
<td>DSN_IDENTITY</td>
</tr>
</tbody>
</table>
Source table column type | Dummy fields in the generated LOAD statement
--- | ---
Row change timestamp column with GENERATED ALWAYS | DSN_RCTIMESTAMP
Row-begin and row-end columns | DSN ROWBEGIN and DSN ROWEND
Transaction-start-ID columns | DSN TRANSACTIONSTID
Generated expression columns | DSN_column-name (where column-name is the name of the column)

### Procedure

To load tables with special column types by using generated LOAD statements

1. Make sure that the target table has a compatible format for the data that you want load.

2. Decide whether you want to include the data for the columns with special column types when you load the unloaded data.

3. If you do not want to load data for the columns with special column types, make sure that the column is defined in the target table as GENERATED ALWAYS.

   The IGNOREFIELDS keyword in the generated LOAD statement causes Db2 to skip the dummy fields when it loads the data into a table.

4. If you want to load data for the columns with special column types, take the following actions:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>For ROWID, identity, or row change timestamp columns:</strong></td>
<td>• In the target table, define the ROWID, identity, or row change timestamp column as GENERATED BY DEFAULT&lt;br&gt;• In the generated LOAD control statement, remove the IGNOREFIELDS keyword and change the dummy field names to the corresponding column names in the target table.</td>
</tr>
<tr>
<td><strong>For row-begin and row-end columns:</strong></td>
<td>• In the target table, define the columns as GENERATED ALWAYS.&lt;br&gt;• Make the following changes to the generated LOAD control statement:&lt;br&gt;  – Specify the OVERRIDE(SYSTEMPERIOD) option.&lt;br&gt;  – Remove the IGNOREFIELDS keyword.&lt;br&gt;  – Change the dummy field names to the corresponding column names in the target table.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| For transaction-start-ID columns: | • In the target table, define transaction-start-ID columns as GENERATED ALWAYS  
• Make the following changes to the generated LOAD control statement:  
  – Specify the OVERRIDE(TRANSID) option.  
  – Remove the IGNOREFIELDS keyword.  
  – Change the dummy field names to the corresponding column names in the target table. |
| For generated expression columns: | Make the following changes to the generated LOAD control statement:  
• Specify the OVERRIDE(NONDETERMINISTIC) option.  
• Remove the IGNOREFIELDS keyword.  
• Change the dummy field names to the corresponding column names in the target table. |

5. Issue the LOAD utility control statement.

**Related concepts:**
“LOAD statements that are generated by UNLOAD” on page 880

**Related reference:**
“Syntax and options of the LOAD control statement” on page 242  
“Syntax and options of the UNLOAD control statement” on page 825

**CREATE TABLE (Db2 SQL)**

**Adding more data to a table or partition**

You might want to use the LOAD utility to add data to a table or partition, rather than replacing existing data.

The RESUME keyword specifies whether data is to be loaded into an empty or a non-empty table space. RESUME NO loads records into an empty table space. RESUME YES loads records into a non-empty table space.

If RESUME NO is specified and the target table is not empty, no data is loaded.

If RESUME YES is specified and the target table is empty, data is loaded.

LOAD always adds rows to the end of the existing rows, but index entries are placed in key sequence.

**Deleting all the data in a table space**

You can use the LOAD utility to efficiently clean out a table space. You can delete all of the data, but retain the structure, including any views and privileges.
Procedure

To delete all the data in a table space:

Submit a LOAD job with the following specifications:

- Specify the REPLACE option in the utility control statement. LOAD REPLACE redefines the table space, but retains all views and privileges that are associated with a table space or table.
- Specify the appropriate LOG value in the utility control statement. If you want this job to be recoverable, specify LOG YES. Otherwise, specify LOG NO so that no rows are logged.
- Specify the input data set in the JCL as DD DUMMY. Such a data set indicates that no rows are to be loaded.

LOAD REPLACE replaces all tables in the table space.

Related reference:

“Syntax and options of the LOAD control statement” on page 242
“Data sets that LOAD uses” on page 307

Loading partitions

You can use the LOAD utility to load one or more partitions of a partitioned table space. To improve performance when loading more than one partition, consider enabling partition parallelism.

About this task

Partition parallelism can reduce the elapsed time that is required for loading large amounts of data into partitioned table spaces.

If you are loading a partitioned table space that is created with DEFINE NO, the load operation might take longer. If a partitioned table space is created with DEFINE NO, all partitions are also implicitly defined with DEFINE NO. The first data row that is inserted by the LOAD utility defines all data sets in the partitioned table space. If this process takes a long time, expect timeouts on the database descriptor (DBD).

Restriction: You cannot load data at the partition level of a partition-by-growth table space.

Procedure

To load partitions:

- If you want to load only certain partitions of a partitioned table, use the PART clause of the INTO TABLE option. If you omit the PART clause, the entire table is loaded.

Restriction: The following restrictions exist for identity columns:

- When index-based partitioning is used, LOAD INTO TABLE PART integer is not allowed if an identity column is part of the partitioning index.
- When table-based partitioning is used, LOAD INTO TABLE PART integer is not allowed if an identity column is used in a partitioning clause of the CREATE TABLE or ALTER TABLE statement.
To override these restrictions, specify the OVERRIDE(IDENTITY) option in the LOAD statement.

- If you want partitions to be processed in parallel, take one of the following actions:
  - If you have a single input data set and the partitioned table space is partitioned (non-UTS) or partition-by-range, specify the PARALLEL keyword. This keyword enables LOAD to use multiple parallel subtasks. When determining the degree of parallelism to specify on the PARALLEL keyword, consider that a high degree of parallelism can result in increased processor time.

  **Recommendation:** Specify PARALLEL(0) or PARALLEL without a number so that Db2 can determine the optimal degree of parallelism.

  - If one or more nonpartitioned secondary indexes exists on the partitioned table space, and you have a separate input data set for each partition, use load partition parallelism. Partition parallelism loads all partitions in a single job. To invoke partition parallelism, for each partition that you want to load, specify the INTO TABLE PART clause with one of the following keywords:
    - INDDN
    - INCURSOR
    - DISCARDDN if DISCARDS n is specified
  
  If the table space is created with DEFINE NO, specifying SHRLEVEL CHANGE on your LOAD statement and enabling partition parallelism is equivalent to concurrent, independent insert jobs. For example, in a large partitioned table space that is created with DEFINE NO, the LOAD utility starts three tasks. The first task tries to insert the first row, which causes an update to the DBD. The other two tasks time out while they wait to access the DBD. The first task holds the lock on the DBD while the data sets are defined for the table space.

  - If the only indexes are the partitioned indexes, use multiple jobs to run LOAD concurrently against separate partitions. This method also requires that you have a separate input data set for each partition.

- If you use the INTO TABLE PART clause, take the following actions as appropriate:
  - If you specify the REPLACE or RESUME options, specify them separately by partition. If you specify these options before the INTO TABLE PART clause, LOAD serializes the load operation for the entire table space and does not process the partitions concurrently.

  - To load columns in an order that is different than the order of the columns in the CREATE TABLE statement, code field specifications for each INTO TABLE PART clause.

  - Make sure that you specify the LOAD options in the appropriate place in the utility statement. Some LOAD options, such as INDDN, DISCARDDN, RESUME, and REPLACE, can have a different scope depending on their placement in the LOAD statement. For example, if you specify INDDN before INTO TABLE, the specified input data set is used to load the entire table. However, if you specify INDDN after INTO TABLE, in a PART clause, the specified input data set is used to load only the specified partition.

- If you want Db2 to save copies of records that are not loaded, use the DISCARDDN option to specify a discard data set as follows:
  - If you want a single discard data set for all partitions, specify DISCARDDN before INTO TABLE. If you use a template for this data set, and the
TEMPLATE utility statement contains the variable $PA. or $PART, that variable is substituted with 0 or the partition number in the first INTO TABLE PART clause.

- If you want one discard data set for a partition, specify DISCARDDN in an INTO TABLE PART clause. If you use a template for this data set, and the TEMPLATE utility statement contains the variable $PA. or $PART, that variable is substituted with the partition number. If you specify DISCARDDN in more than one INTO TABLE PART clause, you must ensure that the data set names are unique. One way to ensure unique names is to use templates and the variable $PA. or $PART in the TEMPLATE statement.

Examples

Example of loading certain records into certain partitions

The control statement in the following example specifies that Db2 is to load data into the first and second partitions of the employee table. Records with '0' in column 1 replace the contents of partition 1; records with '1' in column 1 are added to partition 2; all other records are ignored. This example control statement, which is simplified to illustrate the point, does not list field specifications for all columns of the table.

```sql
LOAD DATA CONTINUEIF(72:72)='X'
  INTO TABLE DSN8B10.EMP PART 1 REPLACE WHEN (1) = '0'
    ( EMPNO  POSITION (1:6) CHAR(6),
      FIRSTNME POSITION (7:18) CHAR(12),
      ...
    )
  INTO TABLE DSN8B10.EMP PART 2 RESUME YES WHEN (1) = '1'
    ( EMPNO  POSITION (1:6) CHAR(6),
      FIRSTNME POSITION (7:18) CHAR(12),
      ...
  )
```

Example of loading partitions from separate input data sets

The following example LOAD statements specify that partitions 1 and 2 of the EMP table are to be loaded from the EMPLDS1 and EMPLDS2 data sets. This example assumes that your data is in separate input data sets and already sorted by partition. Therefore, you do not need to use the WHEN clause of INTO TABLE. Placing the RESUME YES option before the PART option inhibits concurrent partition processing. If you want LOAD to process other partitions concurrently, specify the RESUME option after the PART option.

```sql
LOAD DATA INDDN EMPLDS1 CONTINUEIF(72:72)='X'
    RESUME YES
    INTO TABLE DSN8B10.EMP REPLACE PART 1

LOAD DATA INDDN EMPLDS2 CONTINUEIF(72:72)='X'
    RESUME YES
    INTO TABLE DSN8B10.EMP REPLACE PART 2
```

Example of loading partitions independently

In the following example, partition 1 and partition 2 are loaded concurrently.

```sql
LOAD DATA INDDN SYSREC LOG NO
    INTO TABLE DSN8B10.EMP PART 1 REPLACE

LOAD DATA INDDN SYSREC2 LOG NO
    INTO TABLE DSN8B10.EMP PART 2 REPLACE
```
Example of specifying separate discard data sets for each partition

If you specify multiple discard data sets, you must ensure that the data set names are unique. In the following example, the LOAD statement specifies that two discard data sets are to be allocated, one for each partition. Those discard data sets both use the DT template. The TEMPLATE statement for DT includes the &PA. variable (partition number) to ensure that the data set name is different for each partition.

```
TEMPLATE DT UNIT(SYSDA)
  DSN(JUOSU339.&TS..P&PA..DISCARD)
  SPACE(50,10) TRK
LOAD DATA
  INTO TABLE DSN8B10.EMP PART 1 INDDN D1 DISCARDDN(DT) REPLACE NO RESUME
  INTO TABLE DSN8B10.EMP PART 2 INDDN D2 DISCARDDN(DT) REPLACE NO RESUME
```

Related reference:

“Syntax and options of the LOAD control statement” on page 242
“Syntax and options of the TEMPLATE control statement” on page 793

Loading partition-by-growth table spaces

For partition-by-growth table spaces, the LOAD utility supports loading data only at the table space level, and not at the partition level. If you need additional partitions during the LOAD process and the maximum number of partitions for the table space is not yet reached, the LOAD utility triggers the process to add additional partitions. If the maximum number of partitions is reached, the LOAD utility fails.

Restriction: You cannot use parallelism for LOAD processing for partition-by-growth table spaces.

Related concepts:

Partition-by-growth table spaces (Introduction to Db2 for z/OS)

Loading data containing XML columns

You can load data containing XML columns with one of two methods.

About this task

- The XML column can be loaded from the input record. XML column value can be placed in the INPUT record with or without any other any other loading column values. The input record can be in delimited or non-delimited format. For a non-delimited format, the XML column is treated like a variable character with a 2-byte length preceding the XML value. For a delimited format there are no length bytes present. If the input record is in spanned record format, specify the FORMAT SPANNED YES option.
- The XML column can be loaded from a separate file whether the XML column length is less than 32K or not.

Procedure

To load data into a base table that has XML columns:
1. Create input data sets to ensure that you use the appropriate format:
   - If you use delimited format, specify XML data in the input data set as delimited character strings, separated by the column delimiter.
   - If you do not use delimited format, specify the XML input field length in a 2-byte binary field preceding the data.
2. Create a LOAD utility control statement.
   - To load XML directly from input record, specify XML as the input field type. XML is the only acceptable field type and data type conversion is not supported. Do not specify DEFAULTIF.
   - To load XML from a file, specify CHAR or VARCHAR along with either BLOBF, CLOBF or DBCLOBF to indicate that the input column contains a filename from which a BLOBF, CLOBF or DBCLOBF is to be loaded to the XML column.

3. Submit the utility control statement.

Results

When you load XML documents into a table, and the XML value cannot be cast to the type that you specified when you created the index, the value is ignored without any warnings or errors, and the document is inserted into the table.

When you insert XML documents into a table with XML indexes that are of type DECFLOAT, the values might be rounded when they are inserted. If the index is unique, the rounding might cause duplicates even if the original values are not exactly the same.

Db2 does not compress an XML table space during the LOAD process. If the XML table space is defined with COMPRESS YES, the XML table space is compressed during REORG.

Loading delimited files

You can load a delimited file by using the FORMAT DELIMITED option. A delimited file contains cell values that are separated by delimiters. Delimiters are predefined characters that separate data. The column delimiter separates one column value from the next. Character string delimiters identify the beginning and end of a single cell value and are required only if the cell value contains the column delimiter.

Recommendation: If a delimited file is to be transferred to or from an operating system other than z/OS or between Db2 for z/OS systems that use different EBCDIC or ASCII CCSIDs, use Unicode as the encoding scheme for the delimited file. Using Unicode avoids possible CCSID translation problems.

You are responsible for ensuring that the data in the file does not include the chosen delimiters. If the delimiters are part of the file's data, unexpected errors can occur.

Restrictions: The following restrictions apply to the use of delimiters:
   - You cannot specify the same character for more than one type of delimiter (COLDEL, CHARDEL, and DECPT).
   - If the input data set contains external date, time, or timestamp formats that use separators, the separators cannot be the same character that is specified for a delimiter.
   - You cannot specify a character constant for a delimiter if the utility control statement is not coded in the same encoding scheme as the input file. For example, the utility control statement is coded in Unicode, and the input data is coded in EBCDIC.
You should use the hexadecimal representation for non-default delimiters if the utility control statement is coded in a different encoding scheme than the input file. For example, the utility control statement is coded in Unicode, and the input file is coded in EBCDIC. In this case, if you do not use the hexadecimal representation for the non-default delimiters, the results can be unpredictable.

You do not need to specify the POSITION keyword when you specify the DELIMITED option. The utility ignores the POSITION keyword when you also specify DELIMITED. The utility overrides field data type specifications according to the specifications of the delimited format. (For example, length values for CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, CLOB, DBCLOB, and BLOB data are the delimited lengths of each field in the input data set, and the utility expects all numeric types in external format.)

You cannot specify a binary 0 (zero) for any delimiter.

You cannot specify the default decimal point as a string character delimiter (CHARDEL) or a column string delimiter (COLDEL).

You cannot specify shift-in and shift-out characters for EBCDIC MBCS data.

You cannot specify the pipe character (|) for DBCS data.

You cannot specify the semicolon character (x'5E') as a delimiter character for COLDEL.

You must enclose all space values with CHARDEL if you want to load the character string into a target column that is defined with NOT NULL and without the default value. If a character string is not enclosed by CHARDEL, the utility skips the leading and trailing space characters. If the characters between two column delimiters are all space values, the field is set to null and cannot be loaded into a column.

The following table lists the default hexadecimal values for the delimiter characters based on encoding scheme.

*Table 38. Default delimiter values for different encoding schemes*

<table>
<thead>
<tr>
<th>Character</th>
<th>EBCDIC SBCS</th>
<th>EBCDIC DBCS/MBCS</th>
<th>ASCII/Unicode SBCS</th>
<th>ASCII/Unicode MBCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character string delimiter</td>
<td>X'7F'</td>
<td>X'7F'</td>
<td>X'22'</td>
<td>X'22'</td>
</tr>
<tr>
<td>Decimal point character</td>
<td>X'4B'</td>
<td>X'4B'</td>
<td>X'2E'</td>
<td>X'2E'</td>
</tr>
<tr>
<td>Column delimiter</td>
<td>X'6B'</td>
<td>X'6B'</td>
<td>X'2C'</td>
<td>X'2C'</td>
</tr>
</tbody>
</table>

In most EBCDIC code pages, the hexadecimal values that are specified in the previous table are a double quotation mark ("), a period (.) for the decimal point character, and a comma (,), for the column delimiter.

The following table lists the maximum allowable hexadecimal values for any delimiter character based on the encoding scheme.

*Table 39. Maximum delimiter values for different encoding schemes*

<table>
<thead>
<tr>
<th>Encoding scheme</th>
<th>Maximum allowable value</th>
</tr>
</thead>
<tbody>
<tr>
<td>EBCDIC SBCS</td>
<td>None</td>
</tr>
<tr>
<td>EBCDIC DBCS/MBCS</td>
<td>X'3F'</td>
</tr>
<tr>
<td></td>
<td>(X'7F', X'4B', X'6B' are also allowed.)</td>
</tr>
<tr>
<td>ASCII/Unicode SBCS</td>
<td>None</td>
</tr>
</tbody>
</table>
Table 39. Maximum delimiter values for different encoding schemes  (continued)

<table>
<thead>
<tr>
<th>Encoding scheme</th>
<th>Maximum allowable value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCII/Unicode MBCS</td>
<td>X’7F’</td>
</tr>
</tbody>
</table>

The following table identifies the acceptable data type forms for the delimited file format that the LOAD and UNLOAD utilities use.

Table 40. Acceptable data type forms for delimited files

<table>
<thead>
<tr>
<th>Data type</th>
<th>Acceptable form for loading a delimited file</th>
<th>Form that is created by unloading a delimited file</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR, VARCHAR</td>
<td>A delimited or non-delimited character string</td>
<td>Character data that is enclosed by character delimiters. For VARCHAR, length bytes do not precede the data in the string.</td>
</tr>
<tr>
<td>GRAPHIC (any type)³</td>
<td>A delimited or non-delimited character stream</td>
<td>Data that is unloaded as a delimited character string. For VARGRAPHIC, length bytes do not precede the data in the string.</td>
</tr>
<tr>
<td>INTEGER (any type)¹</td>
<td>A stream of characters that represents a number in EXTERNAL format</td>
<td>Numeric data in external format.</td>
</tr>
<tr>
<td>DECIMAL (any type)²</td>
<td>A character string that represents a number in EXTERNAL format</td>
<td>A string of characters that represents a number.</td>
</tr>
<tr>
<td>DECFLOAT EXTERNAL</td>
<td>A character string that represents</td>
<td>A SQL numeric constant.</td>
</tr>
<tr>
<td>FLOAT ³</td>
<td>A representation of a number in the range -7.2E+75 to 7.2E+75 in EXTERNAL format</td>
<td>A string of characters that represents a number in floating-point notation.</td>
</tr>
<tr>
<td>BLOB, CLOB</td>
<td>A delimited or non-delimited character string</td>
<td>Character data that is enclosed by character delimiters. Length bytes do not precede the data in the string.</td>
</tr>
<tr>
<td>DBCLOB</td>
<td>A delimited or non-delimited character string</td>
<td>Character data that is enclosed by character delimiters. Length bytes do not precede the data in the string.</td>
</tr>
<tr>
<td>DATE</td>
<td>A delimited or non-delimited character string that contains a date value in EXTERNAL format</td>
<td>Character string representation of a date.</td>
</tr>
<tr>
<td>TIME</td>
<td>A delimited or non-delimited character string that contains a time value in EXTERNAL format</td>
<td>Character string representation of a time.</td>
</tr>
</tbody>
</table>
Table 40. Acceptable data type forms for delimited files (continued)

<table>
<thead>
<tr>
<th>Data type</th>
<th>Acceptable form for loading a delimited file</th>
<th>Form that is created by unloading a delimited file</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMESTAMP</td>
<td>A delimited or non-delimited character string that contains a timestamp value in EXTERNAL format</td>
<td>Character string representation of a timestamp.</td>
</tr>
<tr>
<td>TIMESTAMP WITH TIME ZONE</td>
<td>A delimited or non-delimited character string that contains a timestamp with time zone value in EXTERNAL format</td>
<td>Character string representation of a timestamp with time zone.</td>
</tr>
</tbody>
</table>

Note:
1. Field specifications of INTEGER or SMALLINT are treated as INTEGER EXTERNAL.
2. Field specifications of DECIMAL, DECIMAL PACKED, or DECIMAL ZONED are treated as DECIMAL EXTERNAL.
3. Field specifications of FLOAT, REAL, or DOUBLE are treated as FLOAT EXTERNAL.
4. EBCID graphic data must be enclosed in shift-out and shift-in characters.

Related reference:
[Appendix G, “Delimited file format,” on page 1125]

Related information:

Loading data with referential constraints

LOAD does not load a table with an incomplete definition; if the table has a primary key, the unique index on that key must exist. If any table that is to be loaded has an incomplete definition, the LOAD job terminates.

LOAD requires access to the primary indexes on the parent tables of any loaded tables. For simple, segmented, and partitioned table spaces, it drains all writers from the parent table's primary indexes. Other users cannot make changes to the parent tables that result in an update to their own primary indexes. Concurrent inserts and deletes on the parent tables are blocked, but updates are allowed for columns that are not defined as part of the primary index.

By default, LOAD enforces referential constraints, except informational referential constraints, which LOAD ignores. By enforcing referential constraints, LOAD provides you with several possibilities for error:
- Records that are to be loaded might have duplicate values of a primary key.
- Records that are to be loaded might have invalid foreign-key values, which are not values of the primary key of the corresponding parent table.
- The loaded table might lack primary key values that are values of foreign keys in dependent tables.

The next few paragraphs describe how Db2 signals each of those errors and the means it provides for correcting them.

Duplicate values of a primary key

A primary index must be a unique index and must exist if the table definition is complete. Therefore, when you load a parent table, you build at least its primary
You need an error data set, and probably also a map data set and a discard data set.

**Invalid foreign key values:**

A dependent table has the constraint that the values of its foreign keys must be values of the primary keys of corresponding parent tables. By default, LOAD enforces that constraint in much the same way as it enforces the uniqueness of key values in a unique index. First, it loads all records to the table. Subsequently, LOAD checks the validity of the records with respect to the constraints; identifies any invalid record by an error message, and deletes the record from the table. You can choose to copy this record to a discard data set. Again you need at least an error data set, and probably also a map data set and a discard data set.

If a record fails to load because it violates a referential constraint, any of its dependent records in the same job also fail. For example, suppose that the sample project table and project activity tables belong to the same table space, that you load them both in the same job, and that some input record for the project table has an invalid department number. Then, that record fails to be loaded and does not appear in the loaded table; the summary report identifies it as causing a primary error.

However the project table has a primary key, the project number. In this case, the record that is rejected by LOAD defines a project number, and any row in the project activity table that refers to the rejected number is also rejected. The summary report identifies those as causing secondary errors. If you use a discard data set, records for both types of errors are copied to it.

**Missing primary key values**

The deletion of invalid records does not cascade to other dependent tables that are already in place. Suppose now that the project and project activity tables exist in separate table spaces, and that they are both currently populated and possess referential integrity. In addition, suppose that the data in the project table is now to be replaced (using LOAD REPLACE) and that the replacement data for some department was inadvertently not supplied in the input data. Rows that reference that department number might already exist in the project activity table. LOAD, therefore, automatically places the table space that contains the project activity table (and all table spaces that contain dependent tables of any table that is being replaced) into CHECK-pending status.

The CHECK-pending status indicates that the referential integrity of the table space is in doubt; it might contain rows that violate a referential constraint. Db2 places severe restrictions on the use of a table space in CHECK-pending status; typically, you run the CHECK DATA utility to reset this status.

**Consequences of ENFORCE NO**

If you use the ENFORCE NO option, you tell LOAD not to enforce referential constraints. Sometimes you have good reasons for doing that, but the result is that the loaded table space might violate the constraints. Hence, LOAD places the loaded table space in CHECK-pending status. If you use REPLACE, all table spaces that contain any dependent tables of the tables that were loaded are also placed in CHECK-pending status. You must reset the status of each table before you can use any of the table spaces.

Related concepts:
Referential constraint violations

The referential integrity checking in LOAD can delete only incorrect dependent rows, which were input to LOAD. In some circumstances, it is possible to correct referential integrity violations without deleting the dependent rows.

For example, the violations might occur because parent rows do not exist. In this case, correcting the parent tables is better than deleting the dependent rows. In this case, ENFORCE NO is more appropriate than ENFORCE CONSTRAINTS. After you correct the parent table, you can use CHECK DATA to reset the CHECK-pending status.

LOAD ENFORCE CONSTRAINTS is not equivalent to CHECK DATA. LOAD ENFORCE CONSTRAINTS deletes any rows that cause referential constraint violations. CHECK DATA detects violations and optionally deletes such rows. CHECK DATA checks a complete referential structure, although LOAD checks only the rows that are being loaded.

When loading referential structures with ENFORCE CONSTRAINTS, you should load tables before dependent tables.

Compressing data by using the LOAD utility

You can use LOAD to compress data in a table space or partition that is defined with COMPRESS YES. LOAD first builds a compression dictionary. After the dictionary is built, the data is compressed as it is loaded. If a compression dictionary already exists, you can specify that LOAD not build a new one.

Alternatively, you can run the REORG TABLESPACE utility to compress your data.

Before you begin

Your table space or partition in a partitioned table space must be defined with COMPRESS YES before you can compress the data by using LOAD.

Introductory concepts

The effect of data compression on performance (Introduction to Db2 for z/OS)

About this task

LOAD creates a compression dictionary while records are loaded. The data is not compressed until the dictionary is built. To save processing costs, the initial LOAD does not go back to compress the records that were used to build the dictionary. After the dictionary is built, the rest of the data is compressed as it is loaded.

The number of records that are required to build a dictionary depends on the frequency of patterns in the data. For large data sets, a small percentage of the total number of rows is used to build the dictionary. For the best compression results, build a new dictionary whenever you load the data. If a table has DATA CAPTURE CHANGES active, any previously existing dictionary is written to the log.

For partition-by-growth table spaces, the utility builds one dictionary and the same dictionary page is populated through all partitions.
For XML table spaces that are defined with COMPRESS YES, compression does not occur until the first time that the REORG TABLESPACE utility is run on that table space.

**Procedure**

To compress data by using the LOAD utility:

- **If a compression dictionary does not already exist or you want to build a new one**, specify the REPLACE, RESUME NO, or RESUME YES options in the LOAD statement as follows:

  **REPLACE**
  For simple table spaces, you must specify LOAD REPLACE to build new compression dictionaries. If you specify RESUME YES or RESUME NO for a simple table space, LOAD keeps the existing dictionary if one exists. Alternatively, you can run REORG TABLESPACE.

  **RESUME NO**
  If you specify RESUME NO, the table space must be empty.

  **RESUME YES**
  If you specify RESUME YES and SHRLEVEL NONE is explicitly or implicitly specified, LOAD builds a dictionary if the table space is empty. If you specify RESUME YES SHRLEVEL CHANGE, LOAD builds a dictionary when the amount of data in the table space reaches a threshold that is determined by DB2.

- **If you want to use an existing compression dictionary to compress the data**, specify the KEEPDICTIONARY option in the LOAD statement.

  Alternatively, you can run REORG TABLESPACE with the KEEPDICTIONARY option to compress the data. REORG TABLESPACE with KEEPDICTIONARY is efficient, because the data is not decompressed in the process. However, REORG TABLESPACE with KEEPDICTIONARY does not generate a compression report. To get that information, run RUNSTATS to update the catalog statistics and then query the catalog columns yourself.

  One advantage of reusing an existing compression dictionary is that you can eliminate the utility processing time of building the dictionary. Consider specifying KEEPDICTIONARY in the LOAD statement in the following situations:
  - If you are satisfied with the compression that is obtained from an existing dictionary.
  - If the last dictionary was built by REORG TABLESPACE. The REORG TABLESPACE sampling method can yield more representative dictionaries than LOAD and can thus result in a better compression.
  - If the data is not changed significantly since the last dictionary was built.

  You can also specify KEEPDICTIONARY for specific partitions of a partitioned table space. In this case, each partition has its own dictionary.

  The following example LOAD statement reuses an existing compression dictionary:

  ```sql
  LOAD DATA
  REPLACE KEEPDICTIONARY
  INTO TABLE DSN8B10.DEPT
  ( DEPTNO POSITION (1) CHAR(3),
    DEPTNAME POSITION (5) VARCHAR,
  ```
• If you want to copy another compression dictionary from an existing partition into another empty partition, specify the COPYDICTIONARY option in the LOAD statement and use a dummy input data set. In this case, the data that is inserted into the partition is compressed.

What to do next

For XML table spaces, run REORG TABLESPACE on the table space.

Related tasks:

Compressing your data (Db2 Performance)

Related reference:

Chapter 28, “REORG TABLESPACE,” on page 553
Chapter 32, “RUNSTATS,” on page 741

How to load data from DL/I

You can convert data in IMS DL/I databases from a hierarchic structure to a relational structure so that it can be loaded into Db2 tables. To do that, use the DataRefrresher™ and IMS DataPropagator (IMS DPROP) licensed programs.

About this task

You can use DataRefrresher to create source-to-target mappings and to create Db2 databases. After your databases are created and the mappings are set, you can use IMS DPROP to propagate any changes.

IMS DPROP runs as a z/OS application and can extract data from VSAM and physical sequential access method (SAM) files, as well from DL/I databases. Using IMS DPROP, you do not need to extract all the data in a database or data set. You use a statement such as an SQL subselect to indicate which fields to extract and which conditions, if any, the source records or segments must meet.

With JCL models that you edit, you can have IMS DPROP produce the statements for a Db2 LOAD utility job. If you have more than one Db2 subsystem, you can name the one that is to receive the output. IMS DPROP can generate LOAD control statements in the job to relate fields in the extracted data to target columns in Db2 tables.

You have the following choices for how IMS DPROP writes the extracted data:
• 80-byte records, which are included in the generated job stream
• A separate physical sequential data set (which can be dynamically allocated by IMS DPROP), with a logical record length that is long enough to accommodate any row of the extracted data

In the first case, the LOAD control statements that are generated by IMS DPROP include the CONTINUEIF option to describe the extracted data to Db2 LOAD.

In the second case, you can have IMS DPROP name the data set that contains the extracted data in the SYSREC DD statement in the LOAD job. (In that case, IMS DPROP makes no provision for transmitting the extracted data across a network.)
Normally, you do not need to edit the job statements that are produced by IMS DPROP. However, in some cases you might need to edit; for example, if you want to load character data into a Db2 column with INTEGER data type, you need to edit the job statements. (Db2 LOAD does not consider CHAR and INTEGER data to be compatible.)

IMS DPROP is a versatile tool that contains more control, formatting, and output options than are described here. For more information about this tool, see IMS DataPropagator: An Introduction.

### Loading data by using the cross-loader function

The LOAD utility can directly load the output of a dynamic SQL SELECT statement into a table. The dynamic SQL statement can be executed on data at a local server or at any remote server that complies with DRDA. This functionality is called the Db2 family cross-loader function.

#### About this task

This function enables you to use a single LOAD job to transfer data from one location to another location or from one table to another table at the same location. Your input for this cross-loader function can come from other sources besides Db2 for z/OS; you can use IBM Information Integrator Federation feature for access to data from sources as diverse as Oracle and Sybase, as well as the entire Db2 family of database servers.

**Note:** If a table that uses row or column access control security is either the data source or a load target for the cross-loader function, the data is subject to the rules defined in the corresponding row permissions or column masks. The CONTROL column in SYSIBM.SYSTABLES indicates whether row or column access control is activated for a table.

#### Procedure

To load data by using the cross-loader function:

1. Declare a cursor by using the EXEC SQL utility. Within the cursor definition, specify a SELECT statement that identifies the result table that you want to use as the input data for the LOAD job. Use the following rules when writing the SELECT statement:
   - The column names in the SELECT statement must be identical to the column names in the table that is being loaded. You can use the AS clause in the SELECT list to change the column names that are returned by the SELECT statement so that they match the column names in the target table.
   - The columns in the SELECT list do not need to be in the same order as the columns in the target table.
   - The SELECT statement needs to refer to any remote tables by their three-part name.
   - Although you do not need to specify casting functions for any distinct types in the input data or target table, you might need to add casting functions to any additional WHERE clauses in the SQL.

2. Specify the cursor name with the [INCURSOR option](#) in the LOAD statement.
   You can use the same cursor to load multiple tables.
Results

When you submit the LOAD job, Db2 parses the SELECT statement in the cursor definition and checks for errors. If the statement is invalid, the LOAD utility issues an error message and identifies the condition that prevented the execution. If the statement syntax is valid but an error occurs during execution, the LOAD utility also issues an error message. The utility terminates when it encounters an error. If you specify a data-change-table-reference in the from-clause of the cursor, the changes to the source might be committed even though the load fails.

If no errors occur, the utility loads the result table that is identified by the cursor into the specified target table according to the following rules:

- LOAD matches the columns in the input data to columns in the target table by name, not by sequence.
- If the number of columns in the cursor is less than the number of columns in the table that is being loaded, Db2 loads the missing columns with their default values. If the missing columns are defined as NOT NULL without defaults, the LOAD job fails.
- If a source column is defined as NULLABLE and the corresponding target column is defined as NOT NULL without defaults, the LOAD job fails.
- If you specify IGNOREFIELDS YES, LOAD skips any columns in the input data that do not exist in the target table.
- If the data types in the target table do not match the data types in the cursor, Db2 tries to convert the data as much as possible. If the conversion fails, the LOAD job fails. You might be able to avoid these conversion errors by using SQL conversion functions in the SELECT statement of the cursor declaration.
- If the encoding scheme of the input data is different from the encoding scheme of the target table, Db2 converts the encoding schemes automatically. Make sure that the length definition in the target table is able to fit the converted data.
- If the target table has GENERATED ALWAYS columns and you try to load data into them, the LOAD job fails. To avoid this situation, either do not specify these columns in the SELECT statement for the cursor or specify in the LOAD statement that you want to override these columns.

Also, although you do not need to specify casting functions for any distinct types in the input data or target table, you might need to add casting functions to any additional WHERE clauses in the SQL.

Related concepts:

"Before running LOAD” on page 305

Related reference:

“Sample LOAD control statements” on page 362

SYSIBM.SYSTABLES table (Db2 SQL)

How to use inline COPY with LOAD

You can create a full image copy data set (SHRLEVEL REFERENCE) during LOAD execution. The new copy is an inline copy.

About this task

The advantage to using an inline copy is that the table space is not left in COPY-pending status regardless of which LOG option was specified for the utility. Thus, data availability is increased.
**Procedure**

To create an inline copy:

Use the COPYDDN and RECOVERYDDN keywords. You can specify up to two primary and two secondary copies. Inline copies are produced during the RELOAD phase of LOAD processing. You can specify LOAD REPLACE with COPYDDN or RECOVERYDDN or LOAD RESUME YES SHRLEVEL NONE with COPYDDN or RECOVERYDDN. If you specify RESUME NO but not REPLACE, an error message is issued and LOAD terminates.

The SYSCOPY record that is produced by an inline copy contains ICTYPE=F and SHRLEVEL=R. The value of the STYPE column depends on how the image copy was produced:

- R if the image copy was produced by LOAD REPLACE LOG(YES)
- S if the image copy was produced by LOAD REPLACE LOG(NO)
- Y if the image copy was produced by LOAD RESUME LOG(NO)
- Z if the image copy was produced by LOAD RESUME LOG(YES)

The data set that is produced by the inline copy is logically equivalent to a full image copy with SHRLEVEL REFERENCE, but the data within the data set differs in the following ways:

- Data pages might be out of sequence and some might be repeated. If pages are repeated, the last one is always the correct copy.
- Space map pages are out of sequence and might be repeated.
- If the compression dictionary is rebuilt with LOAD, the set of dictionary pages occurs twice in the data set, with the second set being the correct one.

The total number of duplicate pages is small, with a negligible effect on the required space for the data set.

**Related tasks:**

“Replacing data with LOAD” on page 315

**Related reference:**

“COPY-pending status” on page 1040

SYSIBM.SYSCOPY table (Db2 SQL)

---

**Creating a FlashCopy image copy with LOAD**

As part of LOAD processing, you can use FlashCopy technology to take image copies. This method is potentially faster than the traditional Db2 utility methods for creating inline copies and thus reduces the time that data is unavailable. FlashCopy image copies can also potentially reduce the time that is required for recovery operations.

**About this task**

LOAD can also create one to four additional inline image copies by using the traditional methods. Traditional inline image copies are output to a non-VSAM sequential format data set. For more information about creating traditional inline copies, see “How to use inline COPY with LOAD” on page 334.

**Procedure**

To create a FlashCopy image copy with LOAD:
Specify `FLASHCOPY(YES)` or `FLASHCOPY(CONSISTENT)` in the LOAD utility control statement.
Specify `FLASHCOPY(CONSISTENT)` if you specify `SHRLEVEL CHANGE` and want to ensure that the image copy is consistent for recovery purposes. Otherwise, specify `FLASHCOPY(YES)`. Also, be aware that if you specify `CONSISTENT` rather than `YES`, the process of creating an image copy could take longer.

**Restriction:** You cannot specify `CONSISTENT` when copying objects that have been defined with the NOT LOGGED attribute.

As an alternative to specifying `FLASHCOPY` in the LOAD statement, you can set the `FLASHCOPY_LOAD` subsystem parameter to `YES`, which specifies that LOAD is to use `FLASHCOPY(YES)` by default. The value that you specify for the `FLASHCOPY` option in the LOAD statement always overrides the value for the `FLASHCOPY_LOAD` subsystem parameter.

Optionally, you can also specify `FCCOPYDDN` in the LOAD statement. Use this option to specify a template for the FlashCopy image copy. If you do not specify the `FCCOPYDDN` option in the LOAD statement, the utility uses the value from the `FCCOPYDDN` subsystem parameter.

**Restriction:** The data sets that you specify for the FlashCopy image copy must be on FlashCopy Version 2 disk volumes.

When you specify `FLASHCOPY(YES)` or `FLASHCOPY(CONSISTENT)`, LOAD uses FlashCopy technology to create a copy of the table space after the data is loaded. If you also requested one or more inline copies in the LOAD statement (by specifying `REPLACE` and `COPYDDN` or `RECOVERYDDN`), the utility also creates those copies. LOAD does not use the FlashCopy image copy to create those traditional inline copies.

Any indexes that are defined with the `COPY YES` attribute are also copied with FlashCopy technology.
The FlashCopy image copy fails if the FlashCopy Version 2 disk volumes are not available or if any of the other FlashCopy operational restrictions exist. For a list of those operational restrictions, see “FlashCopy image copies” on page 157.

If the FlashCopy copy fails for the target object or auxiliary object and the LOAD statement includes `LOG NO` but not `NOCOPYPEND`, the table space is set to COPY-pending status.

**Related concepts:**
“FlashCopy image copies” on page 157

**Related reference:**
- `[DEFAULT TEMPLATE field (FCCOPYDDN subsystem parameter)](Db2 Installation and Migration)`
- `[LOAD field (FLASHCOPY_LOAD subsystem parameter)](Db2 Installation and Migration)`

---

**Improving LOAD performance**

Depending on the data, target object, and available resources, you can take certain actions that might improve the performance of the LOAD utility. For example, you can preprocess the input data or specify a particular LOAD option.

**About this task**

Be aware that the performance of LOAD on a table that is organized by hash is likely to be slower. The reason is that the rows are loaded according to the hash key rather than sequentially on the pages.
Procedure

To improve LOAD performance, complete one or more of the following recommended actions as appropriate:

- Load numeric data in internal format. *DB2 internal format* is the format that is produced by running UNLOAD with the FORMATINTERNAL option.
- If you specify LOAD REPLACE, specify LOG NO with COPYDDN or RECOVERYDDN to create an inline copy.

- **Preprocess input data:** Present data to LOAD in the optimal order, as follows:
  - Sort the data in cluster order to avoid needing to reorganize it after loading.
  - If you are loading a single table that has, at most, one foreign key or one index key, sort the data in key sequence. (An index over a foreign key is allowed.) If the key is an index key, sort the data in either ascending or descending order, depending on how the index was defined. If the key is a foreign key, sort the data in ascending order. Null key values are treated as “high” values.
  - If you are loading more than one table, choose one of the following methods:
    - Load each table separately. If you use this method, you can follow the rules that are listed in the preceding bullet for loading single tables.
    - Use the WHEN clause under each INTO TABLE option on your LOAD statement to group your input data by table.

Within each table, sort the data in key sequence.

If you do presort the data, specify the PRESORTED YES option in the LOAD statement.

Do any other preprocessing of the input data, as described in “Before running LOAD” on page 305.

- **Avoid conversions:**
  - If possible, avoid data conversion, such as from integer to decimal or from decimal to floating-point.
  - Avoid CCSID and encoding scheme conversions, if possible, by loading data that has the same CCSID as the target table. If you specify a CCSID or encoding scheme option that does not match that of the table that is being loaded, CCSID conversions can occur.

Related information:
- Situations in which character conversion occurs (Db2 Internationalization Guide)

- **Avoid LOB and XML materialization:** To eliminate the need to load large LOBs or XML documents into virtual storage while LOAD is running, use file reference variables for the following data:
  - For XML data that is greater than 32 KB in size.
  - For large LOB data in a row with only 1 LOB. (Large LOBs are usually considered to be 2 MB or greater in size.)

In these cases, LOAD does not materialize LOB or XML data into memory.

- **Use parallelism:** Enable LOAD to use parallelism when possible to reduce the elapsed time for loading large amounts of data. To enable parallelism, take one of the following actions:
  - If the table space is simple, segmented (non-UTS), partitioned (non-UTS), or partition-by-range and all of the data to be loaded is in a single data set, specify the PARALLEL keyword. This keyword enables LOAD to use multiple
parallel subtasks. When you determine the degree of parallelism to specify on
the PARALLEL keyword, consider that a high degree of parallelism can result
in increased processor time. The recommended value is to specify
PARALLEL(0) or PARALLEL without a number so that Db2 can determine
the optimal degree of parallelism.

- If the table space is partitioned and one or more nonpartitioned secondary
  indexes exist, and you have a separate input data set for each partition, use
  partition parallelism. Partition parallelism loads all partitions in a single job.
  To invoke partition parallelism, specify the INTO TABLE PART clause with an
  INDDN specification for each partition.

Alternatively, if you cannot enable parallelism, use multiple jobs to run LOAD
concurrently against separate partitions. This method also requires that you have
a separate input data set for each partition.

If you enable parallelism, allocate extra resources as needed and tune your
system to avoid significant performance degradation. Specifically, to benefit from
parallel operations when you use LOAD SHRLEVEL CHANGE or parallel
inserts, especially when secondary indexes are used, take the following actions:

- Use a larger buffer pool to improve the buffer pool hit ratio.
- Define a higher deferred-write threshold to reduce the number of pages that
  are written to disk, which reduces the I/O time and contention.
- Define a larger checkpoint interval to reduce the number of pages that are
  written to disk, which reduces the I/O time and contention.
- Use ESS Parallel Access Volume (PAV) to support multiple concurrent I/Os to
  the same volume that contains secondary index data sets.
- Use secondary index pieces to support multiple concurrent secondary index
  I/Os.

To fully maximize parallel tasks when loading from a single input data set,
specify the PARALLEL keyword explicitly. If loading from a single input data set
and PARALLEL is not specified, the PARAMDEG_UTIL subsystem parameter
restricts the degree of parallelism for only parallel index build.

- **Use SORTKEYS:** The SORTKEYS option improves performance of the index key
  sort. With SORTKEYS, index keys are passed in memory rather than written to
  work files. Avoiding this I/O to the work files improves LOAD performance.
  LOAD with SORTKEYS also reduces disk space requirements for the SYSUT1
  and SORTOUT data sets, especially if you provide an estimate of the number of
  keys to sort. The SORTKEYS option reduces the elapsed time from the start of
  the RELOAD phase to the end of the BUILD phase.

  The SORTKEYS option is the default if one of the following conditions is true:
  SHRLEVEL is not NONE or SHRLEVEL is NONE, and the target table has one
  or more indexes.

  With SORTKEYS, you can specify an estimate of the number of keys for the job
to sort. If the estimate is omitted or specified as 0, LOAD writes the extracted
  keys to the work data set, which reduces the performance improvement of using
  SORTKEYS.

  To estimate the number of keys to sort:
  1. Count 1 for each index.
  2. Count 1 for each foreign key where the foreign key and index definitions are
     not identical.
  3. For each foreign key where the foreign key and index definitions are
     identical:
     a. Count 0 for the first relationship in which the foreign key participates.
b. Count 1 for subsequent relationships in which the foreign key participates (if any).

4. Multiply the count by the number of rows to be loaded.

If more than one table is being loaded, repeat the preceding steps for each table, and sum the results.

- **Build the indexes in parallel:** You can reduce the elapsed time of a LOAD job for a table space or partition with more than one defined index by having LOAD build the indexes in parallel. See “Building indexes in parallel for LOAD” on page 343.

- **Specify PREFORMAT:** PREFORMAT eliminates the need for Db2 to preformat new pages in a table space during execution time.

  Db2 preformatting sometimes causes a delay, which can affect the performance or execution-time consistency of applications that do many inserts or LOAD jobs with SHRLEVEL CHANGE. When these delays occur and you can predict the table size for a business processing cycle, consider using LOAD PREFORMAT or REORG PREFORMAT. This technique is valuable only when Db2 preformatting causes a measurable delay with processing or causes inconsistent application elapsed times for INSERT operations or LOAD RESUME YES SHRLEVEL CHANGE jobs.

**Recommendation:** Assess performance before and after you use LOAD PREFORMAT or REORG PREFORMAT to quantify its value in your environment.

Using PREFORMAT might eliminate execution-time delays but adds setup time before the application’s execution. The cost of this improvement is an increase in the LOAD or REORG time, because the utility must preformat all pages between the data that is loaded or reorganized and the high-allocated RBA. The additional LOAD or REORG time that is required depends on the amount of disk space that is being preformatted. When this preformatted space is used and Db2 needs to extend the table space, normal data set extending and preformatting occurs.

Consider using preformatting for LOAD SHRLEVEL CHANGE or INSERT processing in the following situations:

- For tables on which many inserts are done and that receive a predictable amount of data. In this case, all of the required space can be preallocated before the application’s execution.
- For a table that acts as a repository for work items that come into a system and that are later used for a backend task that processes the work items.
- For table spaces that start out empty and are populated by many inserts before any query access is run against the table space.

LOAD PREFORMAT or REORG PREFORMAT is not recommended for tables that have a high ratio of reads to inserts if the reads result in table space scans. In this case, preformatting of a table space that contains a table that is used for query processing can cause table space scans to read extra empty pages. This extra reading can extend the elapsed time for these queries.

Mixing inserts and nonindexed queries against a preformatted table space might have a negative impact on the query performance without providing a compensating improvement in the insert performance. Typically, PREFORMAT yields the best results where a high ratio of inserts to read operations exists.

Also, consider the following implications of PREFORMAT on your data sets:

- For user-managed data sets, Db2 does not delete and reallocate them during utility processing. The size of the data set does not shrink back to the original
data set allocation size but either remains the same or increases in size if
more space or data is added. This characteristic has implications when LOAD
or REORG PREFORMAT is used because of the preformatting that is done for
all free pages between the high-used RBA (or page) to the high-allocated
RBA. This preformatting includes secondary extents that have been allocated.

– For Db2-managed data sets, Db2 deletes and reallocates them if you specify
REPLACE on the LOAD or REORG job. This behavior results in the data sets
being resized to their original allocation size. They remain that size if the data
that is being reloaded does not fill the primary allocation and force a
secondary allocation. Therefore, LOAD PREFORMAT or REORG
PREFORMAT with Db2-managed data causes at least the full primary
allocation amount of a data set to be preformatted after the data is reloaded
into the table space.

– For both user-managed and Db2-managed data sets, if the data set goes into
secondary extents during utility processing, the high-allocated RBA becomes
the end of the secondary extent. That value becomes the high value for
preformatting.

Related reference:
“Syntax and options of the LOAD control statement” on page 242

Conversion of input data

The LOAD utility converts data between compatible data types. The source type is
used for user-defined distinct types.

The tables shown below identify the compatibility of data types for assignments
and comparisons. Y indicates that the data types are compatible. N indicates that
the data types are not compatible. D indicates the defaults that are used when you
do not specify the input data type in a field specification of the INTO TABLE
statement.

The following table shows the compatibility of numeric data types.

Table 41. Compatibility of converting numeric data types.

<table>
<thead>
<tr>
<th>Input data types</th>
<th>Output data types</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SMALLINT</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>D</td>
</tr>
<tr>
<td>BIGINT</td>
<td>Y</td>
</tr>
<tr>
<td>INTEGER</td>
<td>Y</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>Y</td>
</tr>
<tr>
<td>FLOAT</td>
<td>Y</td>
</tr>
<tr>
<td>DECFLOAT</td>
<td>Y</td>
</tr>
</tbody>
</table>

Notes:
1. Loading a DECFLOAT or FLOAT column from a DECIMAL PACKED input field can produce unpredictable results. Instead, use
the DECIMAL EXTERNAL format for the input field.

The following table shows the compatibility of character data types.

Table 42. Compatibility of converting character data types.

<table>
<thead>
<tr>
<th>Input data types</th>
<th>Output data types</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BLOB</td>
</tr>
<tr>
<td>CHAR</td>
<td>Y</td>
</tr>
<tr>
<td>CHAR MIXED</td>
<td>Y</td>
</tr>
</tbody>
</table>
### Table 42. Compatibility of converting character data types. (continued)

<table>
<thead>
<tr>
<th>Input data types</th>
<th>BLOB</th>
<th>CHAR</th>
<th>VARCHAR</th>
<th>CLOB</th>
<th>GRAPHIC</th>
<th>VARCHAR</th>
<th>DBCLOB</th>
<th>ROWID</th>
<th>BINARY</th>
<th>VARCHAR</th>
<th>CHAR</th>
<th>VARCHAR</th>
<th>CLOB</th>
<th>GRAPHIC</th>
<th>VARCHAR</th>
<th>DBCLOB</th>
<th>ROWID</th>
<th>BINARY</th>
<th>VARCHAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARCHAR</td>
<td>Y</td>
<td>Y</td>
<td>D</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>VARCHAR MIXED</td>
<td>Y</td>
<td>Y</td>
<td>D</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>GRAPHIC</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>D</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>D</td>
</tr>
<tr>
<td>VAR-GRAPHIC</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>D</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>D</td>
</tr>
<tr>
<td>ROWID</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>D</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>D</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>D</td>
</tr>
<tr>
<td>BINARY</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>D</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>D</td>
</tr>
<tr>
<td>VAR-BINARY</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>D</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>D</td>
</tr>
</tbody>
</table>

**Note:**
1. Conversion applies when either the input data or the target table is Unicode.

---

The following table shows the compatibility of time data types.

### Table 43. Compatibility of converting time data types.

<table>
<thead>
<tr>
<th>Input data types</th>
<th>DATE</th>
<th>TIME</th>
<th>TIMESTAMP</th>
<th>TIMESTAMP WITH TIME ZONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE EXTERNAL</td>
<td>D</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>TIME EXTERNAL</td>
<td>N</td>
<td>D</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>TIMESTAMP EXTERNAL</td>
<td>Y</td>
<td>Y</td>
<td>D</td>
<td>Y</td>
</tr>
<tr>
<td>TIMESTAMP WITH TIME ZONE EXTERNAL</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>D</td>
</tr>
</tbody>
</table>

**Note:**
1. If the data type of the target column is TIMESTAMP WITH TIME ZONE and the timestamp value that is being loaded does not contain a time zone, the LOAD utility uses the value that you specify for the IMPLICIT_TZ option. If you do not specify this option, Db2 uses the value from the IMPLICIT_TIMEZONE DECP value.

Input fields with data types CHAR, CHAR MIXED, CLOB, DBCLOB, VARCHAR, VARCHAR MIXED, GRAPHIC, GRAPHIC EXTERNAL, and VARGRAPHIC are converted from the CCSIDs of the input file to the CCSIDs of the table space when they do not match. For example:

- You specify the ASCII or UNICODE option for the input data, and the table space is EBCDIC.
- You specify the EBCDIC or UNICODE option, and the table space is ASCII.
- You specify the ASCII or EBCDIC option, and the table space is Unicode.
- The CCSID option is specified, and the CCSIDs of the input data are not the same as the CCSIDs of the table space.

CLOB, BLOB, and DBCLOB input field types cannot be converted to any other field type.

Conversion errors cause LOAD:

- To abend, if no discard data set is provided or if the discard limit is exceeded.
- To map the input record for subsequent discarding and continue (if a discard data set is provided)

Truncation of the decimal part of numeric data is not considered a conversion error.

**Related reference:**
Specifying input fields

You can specify input fields in the LOAD utility control statement.

Procedure

To specify input fields in a LOAD utility control statement:

Take one of the following actions:

- Specify the length of VARCHAR, BLOB, CLOB, DBCLOB, ROWID, VARBINARY, TIMESTAMP, and TIMESTAMP WITH TIME ZONE data in the input file.
- Explicitly define all input field specifications.
- Use DECIMAL EXTERNAL(length, scale) in full.
- Specify decimal points explicitly in the input file.

Specifying the TRUNCATE and STRIP options

You can load certain fields that are longer than the length of target column by truncating the data. Db2 truncates the data only when you explicitly specify the TRUNCATE option.

You can specify TRUNCATE with the CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, BINARY, and VARBINARY data type options. LOAD first applies any CCSID conversion, and then truncates the data. The TRUNCATE option of the LOAD utility truncates string data, and it has a different purpose than the SQL TRUNCATE scalar function.

You can also remove a specified character from the beginning, end, or both ends of the data by specifying the STRIP option. This option is valid only with the CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, BINARY, and VARBINARY data type options. If you specify both the TRUNCATE and STRIP options, LOAD performs the strip operation first. For example, if you specify both TRUNCATE and STRIP for a field that is to be loaded into a VARCHAR(5) column, LOAD alters the character strings as shown in the following table. In this table, an underscore represents a character that is to be stripped.

<table>
<thead>
<tr>
<th>Specified STRIP option</th>
<th>Input string</th>
<th>String after strip operation</th>
<th>String that is loaded</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRIP BOTH</td>
<td>'<em>ABCDEFG</em>'</td>
<td>'ABCDEFG'</td>
<td>'ABCDE'</td>
</tr>
<tr>
<td>STRIP LEADING</td>
<td>'<em>ABC</em>'</td>
<td>'ABC_'</td>
<td>'ABC_'</td>
</tr>
<tr>
<td>STRIP TRAILING</td>
<td>'<em>ABC_DEF</em>'</td>
<td>'_ABC_DEF'</td>
<td>'<em>ABC</em>'</td>
</tr>
</tbody>
</table>

How LOAD builds indexes while loading data

LOAD builds all the indexes that are defined for any table that is being loaded.
At the same time the indexes are being built, LOAD checks for duplicate values of any unique index key. If LOAD finds any duplicate values, none of the corresponding rows are loaded. Error messages identify the input records that produce duplicates; optionally, the records are copied to a discard data set. At the end of the job, a summary report lists all errors that are found.

For unique indexes, any two null values are assumed to be equal, unless the index was created with the UNIQUE WHERE NOT NULL clause. In that case, if the key is a single column, it can contain any number of null values, although its other values must be unique.

Neither the loaded table nor its indexes contain any of the records that might have produced an error. Using the error messages, you can identify faulty input records, correct them, and load them again. If you use a discard data set, you can correct the records there and add them to the table with LOAD RESUME.

### Building indexes in parallel for LOAD

Parallel index build reduces the elapsed time for a LOAD job by sorting the index keys and rebuilding multiple indexes in parallel, rather than sequentially. Optimally, a pair of subtasks process each index; one subtask sorts extracted keys while the other subtask builds the index.

LOAD begins building each index as soon as the corresponding sort produces its first sorted record.

LOAD uses parallel index build if all of the following conditions are true:

- At least one index needs to be built.
- The LOAD utility statement specifies a non-zero estimate of the number of keys on the SORTKEYS option.

The degree of parallelism is restricted according to the number of indexes, system resources, and the specification of the PARAMDEG_UTIL subsystem parameter and PARALLEL keyword.

You can either allow the utility to dynamically allocate the data sets that the SORT phase needs, or provide the necessary data sets yourself. Select one of the following methods to allocate sort work and message data sets:

**Method 1:** LOAD determines the optimal number of sort work and message data sets.

1. Specify the SORTDEVT keyword in the utility statement.
2. Allow dynamic allocation of sort work data sets by **not** supplying SORTWKnn DD statements in the LOAD utility JCL.
3. Allocate UTPRINT to SYSOUT.

**Method 2:** You control allocation of sort work data sets, while LOAD allocates message data sets.

1. Provide DD statements with DD names in the form SWnnWKnn.
2. Allocate UTPRINT to SYSOUT.

**Method 3:** You have the most control over rebuild processing; you must specify both sort work and message data sets.

1. Provide DD statements with DD names in the form SWnnWKnn.
2. Provide DD statements with DD names in the form UTPRINnn.

Using this method does not eliminate the requirement for a UTPRINT DD card.

**Data sets used**

If you select Method 2 or 3 in the preceding information, use the following information to define the necessary data sets.

Each sort subtask must have its own group of sort work data sets and its own print message data set. Possible reasons to allocate data sets in the utility job JCL rather than using dynamic allocation are:

- To control the size and placement of the data sets
- To minimize device contention
- To optimally use free disk space
- To limit the number of utility subtasks that are used to build indexes

The DD names SWnnWKnn define the sort work data sets that are used during utility processing. nn identifies the subtask pair, and mm identifies one or more data sets that are to be used by that subtask pair. For example:

**SW01WK01**

The first sort work data set that is used by the subtask as it builds the first index.

**SW01WK02**

The second sort work data set that is used by the subtask as it builds the first index.

**SW02WK01**

The first sort work data set that is used by the subtask as it builds the second index.

**SW02WK02**

The second sort work data set that is used by the subtask as it builds the second index.

The DD names UTPRINnn define the sort work message data sets that are used by the utility subtask pairs. nn identifies the subtask pair.

Every time you invoke LOAD, new UTPRINnn data sets are dynamically allocated. LOAD does not reuse UTPRINnn data sets from previous job steps. This behavior might cause the available JES2 job queue elements to be consumed more quickly than expected.

**Determining the number of sort subtasks**

The maximum number of utility subtask pairs that are started for parallel index build is equal to the number of indexes that are to be built.

LOAD determines the number of subtask pairs according to the following guidelines:

- The number of subtask pairs equals the number of sort work data set groups that are allocated.
- The number of subtask pairs equals the number of message data sets that are allocated.
• If you allocate both sort work and message data set groups, the number of subtask pairs equals the smallest number of data sets that are allocated.

**Allocation of sort subtasks**

The LOAD utility attempts to assign one sort subtask pair for each index that is to be built. If the LOAD utility cannot start enough subtasks to build one index per subtask pair, it allocates any excess indexes across the pairs (in the order that the indexes were created), so that one or more subtask pairs might build more than one index.

During parallel index build processing, LOAD assigns all foreign keys to the first utility subtask pair. Remaining indexes are then distributed among the remaining subtask pairs according to the creation date of the index. If a table space does not participate in any relationships, LOAD distributes all indexes among the subtask pairs according to the index creation date, assigning the first created index to the first subtask pair.

Refer to the following table for conceptual information about subtask pairing when the number of indexes (seven indexes) exceeds the available number of subtask pairs (five subtask pairs).

*Table 45. LOAD subtask pairing for a relational table space*

<table>
<thead>
<tr>
<th>Subtask pair</th>
<th>Assigned index</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW01WKmm</td>
<td>Foreign keys, fifth created index</td>
</tr>
<tr>
<td>SW02WKmm</td>
<td>First created index, sixth created index</td>
</tr>
<tr>
<td>SW03WKmm</td>
<td>Second created index, seventh created index</td>
</tr>
<tr>
<td>SW04WKmm</td>
<td>Third created index</td>
</tr>
<tr>
<td>SW05WKmm</td>
<td>Fourth created index</td>
</tr>
</tbody>
</table>

**Estimating the sort work file size**

If you choose to provide the data sets, you need to know the size and number of keys in all of the indexes that are being processed by the subtask in order to calculate each sort work file size. After you determine which indexes are assigned to which subtask pairs, use one of the following formulas to calculate the required space:

• If the indexes being processed include a mixture of data-partitioned secondary indexes and nonpartitioned indexes, use the following formula: 
  \[
  2 \times (\text{longest index key} + 15) \times (\text{number of extracted keys})
  \]

• Otherwise, if only one type of index is being built, use the following formula: 
  \[
  2 \times (\text{longest index key} + 13) \times (\text{number of extracted keys})
  \]

*longest index key*

The length of the longest key that is to be processed by the subtask. For the first subtask pair for LOAD, compare the length of the longest key and the length of the longest foreign key, and use the larger value. For nonpadded indexes, longest index key means the maximum possible length of a key with all varying-length columns, padded to their maximum lengths, plus 2 bytes for each varying-length column.

*number of extracted keys*

The number of keys from all indexes that are to be sorted and that the subtask is to process.
How LOAD leaves free space

When it loads data into a nonsegmented table space, the LOAD utility leaves one free page after reaching the FREEPAGE limit. This free page is added regardless of whether the loaded records belong to the same or different tables.

When loading into a segmented table space, LOAD leaves free pages, and free space on each page, in accordance with the current values of the FREEPAGE and PCTFREE parameters. (You can set those values with the CREATE TABLESPACE, ALTER TABLESPACE, CREATE INDEX, or ALTER INDEX statements.) LOAD leaves one free page after reaching the FREEPAGE limit for each table in the table space.

For XML table spaces, FREEPAGE and PCTFREE are not processed until the first REORG.

Loading with RECOVER-pending, REBUILD-pending, or REORG-pending status

You cannot load records by specifying RESUME YES if any partition of a table space is in the RECOVER-pending status. In addition, you cannot load records if any index on the table that is being loaded is in the REBUILD-pending status.

If you are replacing a partition, these preceding restrictions are relaxed; the partition that is being replaced can be in the RECOVER-pending status, and its corresponding index partition can be in the REBUILD-pending status. However, all secondary indexes must not be in the page set REBUILD-pending status and KEEPDICTIONARY must not have been specified on active compressed partitions.

The one RECOVER-pending restrictive status has the following description:

RECP RECOVER-pending status is set on a table space or partition. If a single logical partition is in RECP status, the partition is treated as RECP status for SQL access. A single logical partition in RECP status does not restrict utility access to other logical partitions that are not in RECP status. RECP status is reset by recovering only the single logical partition.

The four REBUILD-pending restrictive states have the following descriptions:

RBDP REBUILD-pending status is set on a physical or logical index partition. The individual physical or logical partition is inaccessible and must be rebuilt by using the REBUILD INDEX utility, or recovered by using the RECOVER utility.

PSRBD Page set REBUILD-pending is set on nonpartitioned secondary indexes. Partitioned indexes, including data-partitioned secondary indexes, are never placed in a page set REBUILD-pending status. The entire index space is inaccessible until you rebuild it with the REBUILD utility, or recover it with the RECOVER utility.
REBUILD-pending star status is set only on logical partitions of nonpartitioning indexes. The entire index is inaccessible, but it is made available again when the affected partitions are rebuilt by using the REBUILD INDEX utility, or recovered by using the RECOVER utility.

The one REORG-pending restrictive status has the following description:

**REORP**

REORG-pending status indicates that a table space or partition needs to be reorganized.

Related concepts:

“Resetting the REBUILD-pending status” on page 452

---

**Exit procedures**

Any field procedure that is associated with a column of a table that is being loaded is executed to encode the data before it is loaded. The field procedures for all columns are executed before any edit or validation procedure for the row.

Any field specification that describes the data is checked before a field procedure is executed. That is, the field specification must describe the data as it appears in the input record.

**Loading ROWID columns**

Columns that are defined as ROWID can be designated as input fields; refer to the LOAD field specification syntax diagram. LOAD PART is not allowed if the ROWID column is part of the partitioning key. In this situation, Db2 issues error message DSNU256I.

Columns that are defined as ROWID can be designated as GENERATED BY DEFAULT or GENERATED ALWAYS. With GENERATED ALWAYS, Db2 always generates a row ID.

**ROWID generated by default**

The LOAD utility can load data from input data columns that are defined as ROWID GENERATED BY DEFAULT. The input field must be specified as a ROWID. No conversions are allowed. The input data for a ROWID column must be a unique, valid value for a row ID. If the value of the row is not unique, a duplicate key violation occurs. If such an error occurs, the load fails. In this case, you need to discard the duplicate value and re-run the LOAD job with a new unique value, or allow Db2 to generate the value of the row ID.

You can use the DEFaulTIF attribute with the ROWID keyword. If the condition is met, the column is loaded with a value that is generated by Db2. You cannot use the NULLIF attribute with the ROWID keyword because row ID columns cannot be null.

**Row change timestamp generated always**

The row change timestamp column that is defined as GENERATED ALWAYS cannot be included in the field specification list unless you specify IGNOREFIELDS YES, because Db2 generates the timestamp value for this column.
Loading a LOB column

LOB columns are treated by the LOAD utility as varying-length data. The length value for a LOB column must be 4 bytes.

Procedure

To load a LOB column:

Take one of the following actions:

- **Load the LOB value directly from the input data set**: To load a LOB value directly from the input data set:
  1. In the input data set, include the LOB value preceded by a 4-byte binary field that contains the length of the LOB.
  2. Specify CLOB, BLOB, or DBCLOB in the field specification portion of the LOAD statement. These options indicate that the field in the input data set is a LOB value. For example, to load a CLOB into the RESUME column, specify something like `RESUME POSITION(7) CLOB`. This specification indicates that position 7 of the input data set contains the length of the CLOB followed by the CLOB value that is to be loaded into the RESUME column.
  3. If the input record is in spanned record format, specify `FORMAT SPANNED YES` and specify the LOB fields at the end of the field specification list.

- **Load the LOB value from a file that is listed in the input data set**: When you load a LOB value from a file, the LOB value can be greater than 32 KB. To load a LOB value from a file:
  1. In the input data set, specify the names of the files that contain the LOB values. Each file can be either a PDS, PDSE, or an HFS file.
  2. Specify either BLOBF, CLOBF, or DBCLOBF in the field specification portion of the LOAD statement. For example, to load a LOB into the RESUME column of a table, specify something like `RESUME POSITION(7) VARCHAR CLOBF`. This specification indicates that position 7 of the input data set contains the name of a file from which a varying-length CLOB is to be loaded into the RESUME column.
  3. To insert an empty LOB value into a LOB column, specify one of the following items in the LOAD statement:
     - A blank file name for CHAR CLOBF, CHAR BLOBF, or CHAR DBCLOBF
     - A blank file name for VARCHAR CLOBF, VARCHAR BLOBF, or VARCHAR DBCLOBF
     - A file name with length 0 for VARCHAR CLOBF, VARCHAR BLOBF, or VARCHAR DBCLOB
     Each of these items tell the LOAD utility that the LOB is empty, and the LOAD utility does insert it into the auxiliary table space. LOAD uses a column indicator to indicate that the LOB is empty.
     This step assumes that the LOB is not NULL.

- **Load data from another table**: To transfer data from one location to another location or from one table to another table at the same location, use a cursor. This method of loading data is called the cross-loader function.
  
  When you use the cross-loader function, the LOB value can be greater than 32 KB. For this method, Db2 uses a separate buffer for LOB data and therefore stores only 8 bytes per LOB column. The sum of the lengths of the non-LOB columns plus the sum of 8 bytes per LOB column cannot exceed 32 KB.

Related tasks:
LOAD LOG on a LOB table space

A LOB table space that is defined with LOG YES or LOG NO affects logging during the load of a LOB column.

The following table shows the logging output and LOB table space effect, if any.

Table 46. LOAD LOG and REORG LOG impact for a LOB table space

<table>
<thead>
<tr>
<th>LOAD LOG/ REORG LOG keyword</th>
<th>LOB table space LOG attribute</th>
<th>What is logged</th>
<th>LOB table space status after utility completes</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOG YES</td>
<td>LOG YES</td>
<td>Control information and LOB data</td>
<td>No pending status</td>
</tr>
<tr>
<td>LOG YES</td>
<td>LOG NO</td>
<td>Control information</td>
<td>No pending status</td>
</tr>
<tr>
<td>LOG NO</td>
<td>LOG YES</td>
<td>Nothing</td>
<td>COPY-Pending(^1)</td>
</tr>
<tr>
<td>LOG NO</td>
<td>LOG NO</td>
<td>Nothing</td>
<td>COPY-Pending(^1)</td>
</tr>
</tbody>
</table>

Note:
1. REORG LOG NO of a LOB table space requires SHRLEVEL REFERENCE, which requires that an inline copy be taken during the REORG. This means that you never set COPY-pending for REORG of LOB table spaces under any circumstances.

Loading an XML column

XML columns are treated by the LOAD utility as varying-length data. The length value for an XML column must be 2 bytes.

About this task

LOAD performance can be improved if the input data is in binary XML format.

Procedure

To load an XML column:

Use one of the following approaches:

- **Load the XML value directly from the input data set:** To load an XML value directly from the input data set:
  1. In the input data set, include the XML value preceded by a 2-byte binary field that contains the length of the XML column.
  2. When loading directly from an input record, you must specify XML as the input field type. This is the only acceptable input field type for loading XML column from input record. For example, to load a data into the RESUME column which is XML, specify something like `RESUME POSITION(7) XML`. This specification indicates that position 7 of the input data set contains the length of the XML followed by the XML value that is to be loaded into the RESUME column.

If the input data is in Extensible Dynamic Binary XML Db2 Client/Server Binary XML Format (binary XML format), you need to specify XML BINARXML as the input field type.

3. If the input record is in spanned record format, specify FORMAT SPANNED YES and specify the XML fields at the end of the field specification list.
• **Load the XML value from a file that is listed in the input data set:** When you load an XML value from a file, the XML value can be greater than 32 KB. To load an XML value from a file:

  1. In the input data set, specify the name of the file that contains the value to be loaded to the XML column. The file name can be a PDS, PDSE or a HFS file.

  2. Specify either BLOBF, CLOBF, or DBCLOBF in the field specification portion of the LOAD statement. For example, to load a CLOB file into an XML column RESUME, specify something like RESUME POSITION(7) VARCHAR CLOBF. This specification indicates that position 7 of the input data set contains the name of a file from which a varying-length CLOB is to be loaded into the RESUME column.

    If the input data is in binary XML format, you need to specify BLOBF BINARYXML in the field specification.

  3. When data loaded into an XML column that has an XML type modifier, LOAD validates the input data according to the XML schema that is specified in the type modifier. If LOAD detects a violation, it deletes the row and issues a message to identify the violation.

**Related concepts:**

[Best practices for XML performance in Db2 (Db2 Performance)]

### LOAD LOG on an XML table space

An XML table space that is defined with LOG YES or LOG NO affects logging during the load of an XML column.

The following table shows the logging output and XML table space effect, if any.

**Table 47. LOAD LOG impact for an XML table space**

<table>
<thead>
<tr>
<th>LOAD LOG keyword</th>
<th>XML table space LOG attribute</th>
<th>What is logged</th>
<th>XML table space status after utility completes</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOG YES</td>
<td>LOG YES</td>
<td>Data</td>
<td>No pending status</td>
</tr>
<tr>
<td>LOG YES</td>
<td>LOG NO</td>
<td>Nothing</td>
<td>No pending status</td>
</tr>
<tr>
<td>LOG NO</td>
<td>LOG YES</td>
<td>Nothing</td>
<td>COPY-Pending</td>
</tr>
<tr>
<td>LOG NO</td>
<td>LOG NO</td>
<td>Nothing</td>
<td>ICOPY-Pending</td>
</tr>
</tbody>
</table>

**Running LOAD RESUME YES SHRLEVEL CHANGE without logging**

If you run the LOAD utility with the RESUME YES SHRLEVEL CHANGE options, you can improve the performance by limiting the logging that is done by Db2 during the LOAD operation.

**Procedure**

To run LOAD RESUME YES SHRLEVEL CHANGE without logging:

1. Alter the table space to NOT LOGGED.
2. Run the LOAD job with the RESUME YES SHRLEVEL CHANGE options.

   If the LOAD job fails, you cannot restart it, because the target is a NOT LOGGED table space. Instead terminate the job, recover the data from a previous image copy, and rerun the LOAD job.
3. Alter the table space back to LOGGED.
4. Take an image copy of the table space.

**Related concepts:**

- The NOT LOGGED attribute (Db2 Administration Guide)

**Related reference:**

- ALTER TABLESPACE (Db2 SQL)

**Related information:**

---

### Collecting inline statistics while loading a table

If you do not specify LOAD RESUME YES, you can use the STATISTICS keyword to gather inline statistics. In most cases, using the STATISTICS keyword eliminates the need to run RUNSTATS after loading a table space.

However, if you perform a LOAD PART operation, you should run RUNSTATS INDEX on the nonpartitioned secondary indexes to update the catalog data about these indexes.

**Procedure**

To collect statistics while loading a table:

1. Use the STATISTICS option to collect statistics so that the Db2 catalog statistics contain information about the newly loaded data:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collecting inline statistics for discarded rows</td>
<td>If you specify the DISCARDDN option when you collect inline statics and a row is found with check constraint errors or conversion errors, the row is not loaded into the table. However, the LOAD utility collects inline statistics before it identifies the rows to discard. As a general rule, if the number of discarded rows is larger than 20 percent of the total number of rows in the table, run the RUNSTATS utility separately on the table after running LOAD.</td>
</tr>
<tr>
<td>Collecting inline statistics for data partitioned secondary indexes</td>
<td>To collect inline statistics on data partitioned secondary indexes, you must allocate sort work data sets.</td>
</tr>
</tbody>
</table>

If you perform a LOAD operation on a base table that contains an XML column, Db2 does not collect inline statistics for the related XML table space or its indexes. Recording these new statistics enables Db2 to select SQL paths with accurate information.

2. Rebind any application plans that depend on the loaded tables to update the path selection of any embedded SQL statements.

**What to do next**

To collect statistics on the loaded table, you might need to invoke the RUNSTATS utility after the LOAD utility processing has completed.

**Related tasks:**
Inline COPY for a base table space

If you take an inline image copy of a table that has LOB columns, Db2 makes a copy of the base table space, but does not copy the LOB table spaces.

Termination of LOAD

The effects of terminating a LOAD utility job depends on the phase the utility was in when it was terminated.

If you terminate LOAD by using the TERM UTILITY command during the reload phase, the records are not erased. The table space remains in RECOVER-pending status, and indexes remain in the REBUILD-pending status.

If you terminate LOAD by using the TERM UTILITY command during the sort or build phases, the indexes that are not yet built remain in the REBUILD-pending status.

If you terminate a LOAD SHRLEVEL CHANGE, uncommitted records are rolled back, but committed records remain in the table. The table space is not in RECOVER-pending status, and the indexes are not in REBUILD-pending status.

If the LOAD job terminates during the RELOAD, SORT, BUILD, or SORTBLD phases, both RESTART and RESTART(PHASE) phases restart from the beginning of the RELOAD phase. However, restart of LOAD RESUME YES or LOAD PART RESUME YES in the BUILD or SORTBLD phase results in message DSNU257I.

If the LOAD job terminates during the DISCARD phase, a message is issued to indicate that the LOAD job was terminated, an RC=8 is issued, and discard processing completes.

The following table lists the LOAD phases and their effects on any pending states when the utility is terminated in a particular phase.

Table 48. LOAD phases and their effects on pending states when terminated.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Effect on pending status</th>
</tr>
</thead>
</table>
| Reload     | •Places table spaces in RECOVER-pending status, and then resets the status if there are no unique indexes.  
             | •Places indexes in REBUILD-pending status.  
             | •Places table spaces in COPY-pending status if there are no unique indexes.  
             | •Places table spaces in CHECK-pending status. |
| Build      | •Resets REBUILD-pending status for non-unique indexes.  
             | •Resets RECOVER-pending status for table spaces with unique indexes, if no INDEXVAL phase is needed.  
             | •Places table spaces in COPY-pending status. |
Table 48. LOAD phases and their effects on pending states when terminated. (continued)

<table>
<thead>
<tr>
<th>Phase</th>
<th>Effect on pending status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indexval</td>
<td>• Resets REBUILD-pending status for unique indexes.</td>
</tr>
<tr>
<td></td>
<td>• Resets RECOVER-pending status for table spaces with unique indexes.</td>
</tr>
<tr>
<td></td>
<td>• Places table spaces in COPY-pending status.</td>
</tr>
<tr>
<td>Enforce</td>
<td>• Resets CHECK-pending status for table spaces.</td>
</tr>
</tbody>
</table>

**Restart of LOAD**

You can restart a LOAD utility job.

You can restart the job either at its last commit point (RESTART(CURRENT)) or at the beginning of the phase during which operation ceased (RESTART(PHASE)). LOAD output messages identify the completed phases. Use the DISPLAY command to identify the specific phase during which operation stopped.

By default, Db2 uses RESTART(CURRENT), except if LOAD is restarting during the UTILINIT phase or the UTILTERM phase. In both of these situations, Db2 uses RESTART(PHASE) by default. You can override the default RESTART values by using the RESTART parameter.

**Restrictions:** The following restrictions apply to restarting LOAD jobs:
- If LOAD abnormally terminates or a system failure occurs while LOAD is in the UTILTERM phase, you must restart with RESTART(PHASE).
- If you restart a LOAD job with the RESUME YES and SORTKEYS NO options for a table that has LOB columns, you must use RESTART(CURRENT).
- If you use RESTART(PHASE) to restart a LOAD job that specified RESUME NO, the LOB table spaces and indexes on auxiliary tables are reset.
- For a table that has LOB columns, you cannot restart a LOAD job that uses the INCURSOR option.
- If you restart a LOAD job that uses the STATISTICS keyword, inline statistics collection does not occur. To update catalog statistics, run the RUNSTATS utility after the restarted LOAD job completes.
- If you are using a BatchPipes file, you cannot restart the LOAD utility. If the application that populates the BatchPipes file terminates, you need to terminate the job where LOAD is running. If the LOAD utility was invoked from a stored procedure, you also need to terminate the WLM application environment of the LOAD utility that reads the BatchPipes file. After you terminate the job, terminate the LOAD utility by using the Db2 TERM UTILITY command, and then you can resubmit the LOAD job.
- You cannot restart LOAD with RESUME and with PRESORTED YES in the RELOAD phase. If you do so, utility processing abnormally terminates, and LOAD issues an error message. You must:
  - Terminate LOAD
  - Recover the table space that is being loaded
  - Recover all indexes on the table space that are in the REBUILD-pending state
- If the LOAD statement includes the PARALLEL option with a value other than 1, you cannot use RESTART(CURRENT); RESTART(PHASE) is used instead.

The following table provides information about restarting LOAD, depending on the phase that LOAD was in when the job stopped. The TYPE column
distinguishes between the effects of specifying RESTART or RESTART(PHASE). Additional phase restrictions are explained in the notes.

Table 49. LOAD restart information

<table>
<thead>
<tr>
<th>Phase</th>
<th>Type of RESTART</th>
<th>Required data sets</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELOAD</td>
<td>CURRENT</td>
<td>SYSREC and SYSUT1 SYSMAP and SYSERR</td>
<td>1, 2, 10</td>
</tr>
<tr>
<td>PHASE</td>
<td>SYSREC</td>
<td></td>
<td>3, 10, 11</td>
</tr>
<tr>
<td>SORT</td>
<td>CURRENT</td>
<td>SYSUT1</td>
<td>4, 10</td>
</tr>
<tr>
<td>PHASE</td>
<td>SYSUT1</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>BUILD</td>
<td>CURRENT</td>
<td>SORTOUT</td>
<td>4, 5, 10</td>
</tr>
<tr>
<td>PHASE</td>
<td>SORTOUT</td>
<td></td>
<td>5, 10</td>
</tr>
<tr>
<td>SORTBLD</td>
<td>CURRENT</td>
<td>SYSUT1 and SORTOUT</td>
<td>5, 6, 10</td>
</tr>
<tr>
<td>PHASE</td>
<td>SYSUT1 and SORTOUT</td>
<td></td>
<td>5, 6, 10</td>
</tr>
<tr>
<td>INDEXVAL</td>
<td>CURRENT</td>
<td>SYSERR or SYSUT1</td>
<td>2</td>
</tr>
<tr>
<td>PHASE</td>
<td>SYSERR or SYSUT1</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>ENFORCE</td>
<td>CURRENT</td>
<td>SORTOUT and SYSUT1</td>
<td>7</td>
</tr>
<tr>
<td>PHASE</td>
<td>SORTOUT and SYSUT1</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>DISCARD</td>
<td>CURRENT</td>
<td>SYSMAP and SYSERR SORTOUT and SYSUT1</td>
<td>7, 8</td>
</tr>
<tr>
<td>PHASE</td>
<td>SYSMAP and SYSERR SORTOUT and SYSUT1</td>
<td></td>
<td>7, 8</td>
</tr>
<tr>
<td>REPORT</td>
<td>CURRENT</td>
<td>SYSERR or SORTOUT SYSMAP and SYSUT1</td>
<td>7, 9</td>
</tr>
<tr>
<td>PHASE</td>
<td>SYSERR or SORTOUT SYSMAP and SYSUT1</td>
<td></td>
<td>7, 9</td>
</tr>
</tbody>
</table>

Notes:
1. SYSMAP and SYSERR data sets might not be required for all LOAD jobs.
2. If the SYSERR data set is not required and not provided, LOAD uses SYSUT1 as a work data set to contain error information.
3. You must not restart during the RELOAD phase if you specified SYSREC DD *. This statement prevents internal commits from being taken, and RESTART performs like RESTART(PHASE), except without data back out. Also, you must not restart if your SYSREC input consists of multiple concatenated data sets.
4. The utility can be restarted with either RESTART or RESTART(PHASE). However, because this phase does not take checkpoints, RESTART is always re-executed from the beginning of the phase.
5. A LOAD RESUME YES job cannot be restarted in the BUILD or SORTBLD phase.
6. Use RESTART or RESTART(PHASE) to restart at the beginning of the RELOAD phase.
7. This utility can be restarted with either RESTART or RESTART(PHASE). However, the utility can be re-executed from the last internal checkpoint. This behavior is dependent on the data sets that are used and whether any input data sets were rewritten.
8. The SYSUT1 data set is required if the target table space is segmented or partitioned.
9. If a report is required and this LOAD job does not specify discard processing, SYSMAP is required to complete the report phase.
10. Any job that finished abnormally in the RELOAD, SORT, BUILD, or SORTBUILD phase and has SORTKEYS enabled restarts from the beginning of the RELOAD phase. (A job that has SORTKEYS enabled means that in the LOAD statement, SORTKEYS was either explicitly specified with a valid value or implicitly specified as the default. In other words, SORTKEYS NO was not specified.)
11. LOAD with RESUME and with PRESORTED YES cannot be restarted in the RELOAD phase.
After running LOAD

After you run the LOAD utility, you might need take additional actions to make sure your objects are consistent and not in restricted status.

Copying the loaded table space or partition

If you ran the LOAD utility with the LOG YES option, consider taking a full image copy of the table space or partition that you loaded. Such a copy might reduce the processing time of subsequent recovery operations.

About this task

If you took primary and backup inline copies during the load operation, you do not need to take full image copies of the table space or partition after LOAD completes. However, you might need to take images copies of indexes.

Procedure

To copy the loaded table space or partition:

Use the COPY utility to create a full image copy. If you specified the RESUME NO option or the REPLACE option for LOAD, take two or more full image copies.

Resetting restricted status after running the LOAD utility

The LOAD utility can place a table space or an index space into a restricted status. Several types of restricted status are possible.

Before you begin

You can issue the following command to discover which spaces are in a restricted status:

- DISPLAY DATABASE (*) SPACENAM (*) RESTRICT

About this task

Your use of a table space in restricted status is severely limited. In general, you cannot access the data through SQL. You can only drop the table space or one of its tables, or perform some operation that resets the status.
Related reference:

Appendix B, “Advisory or restrictive states,” on page 1037

DISPLAY DATABASE (Db2) (Db2 Commands)

Resetting COPY-pending status

If you load with LOG NO and do not take an inline copy, LOAD places a table space in the COPY-pending status. Immediately after that operation, Db2 cannot recover the table space (although you can, by loading it again).

Prepare for recovery, and turn off the restriction, by making a full image copy using SHRLEVEL REFERENCE. (If you end the copy job before it is finished, the table space is still in COPY-pending status.)

You can also remove the restriction by using one of these operations:
- LOAD REPLACE LOG YES
- LOAD REPLACE LOG NO with an inline copy
- REORG LOG YES
- REORG LOG NO with an inline copy
- REPAIR SET with NOCOPYPEND

If you use LOG YES and do not make an image copy of the table space, subsequent recovery operations are possible but take longer than if you had made an image copy.

A table space that is in COPY-pending status can be read without restriction; however, it cannot be updated.

Resetting REBUILD-pending status

LOAD sets index spaces to REBUILD-pending status when a REBUILD job ends before the INDEXVAL phase is complete.

LOAD places all the index spaces for a table space in the REBUILD-pending status if you end the job (by using the TERM UTILITY command) before it completes the INDEXVAL phase. Db2 places the table space in RECOVER-pending status if you end the job before the job completes the RELOAD phase.

Resetting the RECOVER-pending status depends on when the utility terminated:
- If the data is intact and you have a full image copy of the affected indexes, you can recover the indexes using the RECOVER INDEX utility. Run the DISPLAY DATABASE command and examine the output. Data is intact when the output indicates that the indexes are in REBUILD-pending status and the table space is not in RECOVER-pending status. If you do not have an image copy available, you must rebuild the entire index by using the REBUILD INDEX utility. However, for partitioning indexes and for secondary indexes that are in REBUILD-pending (RBDP) status, you can use the PART option of REBUILD INDEX to rebuild separate partitions of the index.
- If the data is not intact, you can either load the table again or recover it to a prior point of consistency. Run the DISPLAY DATABASE command and examine the output. The recovery puts the table space into COPY-pending status and places all indexes in REBUILD-pending status.

CHECK-pending status after running LOAD

The LOAD utility places a table space in CHECK-pending (CHKP) status if its referential integrity is in doubt or its check constraints are violated.
To reset this status, run the CHECK DATA utility, which locates invalid data and, optionally, removes it. If CHECK DATA removes the invalid data, the remaining data satisfies all check and referential constraints and therefore, the CHECK-pending restriction is removed.

Although CHECK DATA is usually the preferred method for resetting CHECK-pending status, you can also reset this status by using any of the following operations:
- Drop tables that contain invalid rows.
- Replace the data in the table space by using LOAD REPLACE and enforcing check and referential constraints.
- Recover all members of the table space that were set to a prior quiesce point.
- Use the REPAIR utility with the SET STATEMENT and NOCHECKPEND option.

**Example of CHKP status after LOAD REPLACE**

Suppose that you replace the contents of the sample project table (PROJ) by using LOAD with the REPLACE option. You also specify that LOAD is to enforce the table's referential and table check constraints. When the LOAD job completes, the project table contains only valid records and is not in CHECK-pending status. However, its dependent table, the project activity table (PROJACT), is placed in CHECK-pending status. Some of its rows might have project numbers that no longer exist in the project table. (If the project table had any other dependents, they also would be in CHECK-pending status.)

In this case, run CHECK DATA against the table space that contains the project activity table to reset the status. When you run CHECK DATA, ensure that all related table spaces are available. Related table spaces means all table spaces that contain either parent tables or dependent tables of any table in the table space that is being checked.

Also, in the CHECK DATA statement, consider the value that you specify for the DELETE option as follows:

**YES**

DELETE YES deletes invalid records and resets the status, but it is not the default value. If you specify DELETE YES, you use another table that is called an exception table to receive copies of the invalid records. For example, if table Y is the exception table for table X, name it with the following clause in the CHECK DATA statement:

```
FOR EXCEPTION IN X USE Y
```

Deletes by CHECK DATA are not subject to any of the SQL delete rules; they cascade without restraint to the lowest-level descendant. Therefore, if you use DELETE YES, you must name an exception table for every descendant of every table in every table space that is being checked.

**NO**

Use DELETE NO, the default value, to quickly determine the size of the problem. If many rows are affected, you can choose to correct the problem by reloading the data rather than using CHECK DATA to correct the current situation.

**Example:**
In the following example, CHECK DATA is to be run against the table space that contains the project activity table (PROJACT). Assume that the exception tables DSN8B10.EPROJACT and DSN8B10.EEPA exist.

```
CHECK DATA TABLESPACE DSN8D11A.PROJACT
DELETE YES
FOR EXCEPTION IN DSN8B10.PROJACT USE DSN8B10.EPROJACT
IN DSN8B10.EMPPROJACT USE DSN8B10.EEPA
SORTDEV'T SYSDA
SORTNUM 4
```

If the statement does not name error or work data sets, the JCL for the job must contain DD statements similar to the following DD statements:

```
//SYSERR DD UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSUT1 DD UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SORTOUT DD UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//UTPRINT DD SYSOUT=A
```

**Example of CHKP status after LOAD RESUME**

Suppose that you want to add records to both the sample project (PROJ) and project activity (PROJACT) tables by using LOAD with the RESUME option. Furthermore, you want to run both jobs at the same time, which you can do because the tables belong to separate table spaces. When you load the project activity table, you must use the ENFORCE NO option in the LOAD statement because you cannot assume that the parent project table is already loaded.

After the two jobs are complete, the table spaces are in the following states:

- If you enforced constraints when loading the project table, the table space is not in CHECK-pending status.
- Because you did not enforce constraints on the project activity table, the table space is in CHECK-pending status.
- Because you used LOAD RESUME (not LOAD REPLACE) when loading the project activity table, its dependents (the employee-to-project-activity table) are not in CHECK-pending status. That is, the LOAD operation did not delete any parent rows from the project activity table, and therefore cannot violate the referential integrity of its dependent.

Therefore, you should run CHECK DATA on the project activity table. If you specify DELETE YES, you need an exception table for PROJACT and an exception table for EMPPROJACT, which is dependent on PROJACT.

To speed CHECK DATA processing, specify the SCOPE PENDING option to limit the checking.

Db2 records the identifier of the first row of the table that might violate referential or table check constraints. For partitioned table spaces, that identifier is in SYSIBM.SYSTABLEPART; for nonpartitioned table spaces, that identifier is in SYSIBM.SYSTABLES.

**Example:**

In the following example, CHECK DATA is to be run against the table space that contains the project activity table after LOAD RESUME:

```
CHECK DATA TABLESPACE DSN8D11A.PROJACT
SCOPE PENDING
DELETE YES
```
As before, the JCL for the job needs DD statements to define the error and sort data sets.

Related reference:

- “CHECK-pending status” on page 1039
- “Syntax and options of the CHECK DATA control statement” on page 74
- “Exception tables for the CHECK DATA utility” on page 92
- “Data sets that CHECK DATA uses” on page 86

Related information:

**Running CHECK INDEX after loading a table that has indexes**

The CHECK INDEX utility tests whether an index is consistent with the data it indexes and issues error messages if it finds an inconsistency.

**About this task**

If you have any reason to doubt the accuracy of an index (for example, if the result of an SQL SELECT COUNT statement is inconsistent with RUNSTATS output) you might want to check the index.

**Procedure**

To check the accuracy of the index:

Invoke the CHECK INDEX utility. You might also want to invoke the CHECK INDEX utility after any LOAD operation that shows some abnormal condition in its execution, or even run it periodically to verify the accuracy of important indexes.

**What to do next**

To rebuild an index that is inconsistent with its data, invoke the REBUILD INDEX utility.

Related reference:

- Chapter 12, “CHECK INDEX,” on page 103
- Chapter 25, “REBUILD INDEX,” on page 427
- Chapter 32, “RUNSTATS,” on page 741

**Recovering data after a failed LOAD job**

If a LOAD utility job fails, you can recover the data to a point in time before the LOAD job ran.
About this task

When you specify LOG YES in the LOAD utility control statement, Db2 inserts a record into the SYSIBM.SYSCOPY catalog table at the beginning of the RELOAD phase of LOAD processing. Db2 uses this SYSCOPY record to help facilitate recovery in case of failure. However, because of this SYSCOPY record, if the LOAD LOG YES job fails, recover to the point in time before the LOAD job was run. Although you can recover the data to the current state, the results are unpredictable.

Procedure

To recover a failed LOAD job, take one of the following actions:

- If the LOAD statement included the LOG YES option, recover the data to a point in time before the LOAD job ran. You can use the RECOVER utility with the TORBA option or another point-in-time recovery option.
- If the LOAD statement included the LOG NO option, recover the data to the point in time before the LOAD job ran or to the current state.

Related concepts:

- “Point-in-time recovery” on page 495
- Options for restoring data to a prior point in time (Db2 Administration Guide)

Related reference:

- SYSIBM.SYSCOPY table (Db2 SQL)

Reorganization of an auxiliary index after LOAD

Indexes on the auxiliary tables are not built during the BUILD phase. Instead, LOB values are inserted (not loaded) into auxiliary tables during the RELOAD phase as each row is loaded into the base table. Each index on the auxiliary table is also updated as part of the insert operation.

Because the LOAD utility inserts keys into an auxiliary index, free space within the index might be consumed and index page splits might occur. Consider reorganizing an index on the auxiliary table after LOAD completes to introduce free space into the index for future inserts and loads.

Effects of running LOAD

The effects of running LOAD can be different, depending on your situation. Running the LOAD utility can have effects on index version numbers, control intervals, and table spaces that are defined with the NOT LOGGED attribute.

The effect of LOAD on index version numbers

Db2 stores the range of used index version numbers in the OLDEST_VERSION and CURRENT_VERSION columns of the following catalog tables:

- SYSIBM.SYNSINDEXES
- SYSIBM.SYNSINDEXPART

The OLDEST_VERSION column contains the oldest used version number, and the CURRENT_VERSION column contains the current version number.
When you run LOAD with the REPLACE option, the utility updates this range of used version numbers for indexes that are defined with the COPY NO attribute. LOAD REPLACE sets the OLDEST_VERSION column to the current version number, which indicates that only one version is active; Db2 can then reuse all of the other version numbers.

Recycling of version numbers is required when all of the version numbers are being used. All version numbers are being used when one of the following situations is true:

- The value in the CURRENT_VERSION column is less than the value in the OLDEST_VERSION column.
- The value in the CURRENT_VERSION column is 15, and the value in the OLDEST_VERSION column is 0 or 1.

You can also run REBUILD INDEX, REORG INDEX, or REORG TABLESPACE to recycle version numbers for indexes that are defined with the COPY NO attribute. To recycle version numbers for indexes that are defined with the COPY YES attribute or for table spaces, run MODIFY RECOVERY.

### The effect of LOAD on table space version numbers

When you run LOAD REPLACE or LOAD RESUME NO, the table space is reset to a single version. Db2 sets the OLDEST_VERSION column value to the CURRENT_VERSION column value in SYSIBM.SYSTABLESPACE. For each table that is in version 0 format, a system pages is inserted in the table space.

For LOAD REPLACE or LOAD RESUME NO, if a system page is missing for a table in the table space, the utility adds a system page for the current version of the table.

For LOAD RESUME YES, if a system page is missing for a table that is in version 0 format, the utility adds a system page for version 0 of the table.

### The effect of LOAD REPLACE on the control interval

When you run a LOAD job with the REPLACE option but without the REUSE option and the data set that contains the data is Db2-managed, Db2 deletes this data set before the LOAD and redefines a new data set with a control interval that matches the page size.

### The effect of LOAD on table spaces defined with NOT LOGGED attribute

The following table shows the effect of LOAD table spaces defined with the NOT LOGGED attribute.
Table 50. LOAD parameters

<table>
<thead>
<tr>
<th>LOAD REORG LOG keyword</th>
<th>Table space logging attribute</th>
<th>Table space type</th>
<th>What is logged</th>
<th>Table space status after utility completes</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOG YES</td>
<td>NOT LOGGED</td>
<td>Non-LOB</td>
<td>LOG YES changes to LOG NO</td>
<td>No pending status or ICOPY-pending¹</td>
</tr>
<tr>
<td>LOG YES</td>
<td>NOT LOGGED</td>
<td>LOB</td>
<td>control information</td>
<td>No pending status</td>
</tr>
<tr>
<td>LOG NO</td>
<td>NOT LOGGED</td>
<td>Non-LOB</td>
<td>nothing</td>
<td>No pending status or ICOPY-pending¹</td>
</tr>
<tr>
<td>LOG NO</td>
<td>NOT LOGGED</td>
<td>LOB</td>
<td>nothing</td>
<td>No pending status</td>
</tr>
</tbody>
</table>

Note:
1. The table space is set to ICOPY-pending status if the records are discarded and no pending status if the records are not discarded.

Related concepts:

Table space versions (Db2 Administration Guide)

Sample LOAD control statements

Use the sample control statements as models for developing your own LOAD control statements.

Example 1: Specifying field positions

The control statement specifies that the LOAD utility is to load the records from the data set that is defined by the SYSREC DD statement into table DSN8810.DEPT. SYSREC is the default input data set.

Each POSITION clause specifies the location of a field in the input record. In this example, LOAD accepts the input that is shown in Figure 25 on page 363 and interprets it as follows:

- The first 3 bytes of each record are loaded into the DEPTNO column of the table.
- The next 36 bytes, including trailing blanks, are loaded into the DEPTNAME column.
  - If this input column were defined as VARCHAR(36), the input data would need to contain a 2-byte binary length field preceding the data. This binary field would begin at position 4.
- The next three fields are loaded into columns that are defined as CHAR(6), CHAR(3), and CHAR(16).

The RESUME YES clause specifies that the table space does not need to be empty; new records are added to the end of the table.
Figure 24. Example of a LOAD statement that specifies field positions

```
LOAD DATA
RESUME YES
INTO TABLE DSN8B10.DEPT
(DEPTNO  POSITION (1:3)  CHAR(3),
 DEPTNAME  POSITION (4:39)  CHAR(36),
 MGRNO  POSITION (40:45)  CHAR(6),
 ADMRDEPT  POSITION (46:48)  CHAR(3),
 LOCATION  POSITION (49:64)  CHAR(16))
```

Figure 25 shows the input to the preceding LOAD job.

```
A00 SPIFFY COMPUTER SERVICE DIV.  000010A00USIBMSTODB21
B01 PLANNING  000020A00USIBMSTODB21
C01 INFORMATION CENTER  000030A00USIBMSTODB21
D01 DEVELOPMENT CENTER  A00USIBMSTODB21
```

Figure 25. Records in an input data set for LOAD

The following table shows the result of executing the statement `SELECT * FROM DSN8B10.DEPT` after the preceding input records are loaded.

**Table 51. Data that is loaded into a table**

<table>
<thead>
<tr>
<th>DEPTNO</th>
<th>DEPTNAME</th>
<th>MGRNO</th>
<th>ADMRDEPT</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>A00</td>
<td>SPIFFY COMPUTER SERVICE DIV.</td>
<td>000010</td>
<td>A00</td>
<td>USIBMSTODB21</td>
</tr>
<tr>
<td>B01</td>
<td>PLANNING</td>
<td>000020</td>
<td>A00</td>
<td>USIBMSTODB21</td>
</tr>
<tr>
<td>C01</td>
<td>INFORMATION CENTER</td>
<td>000030</td>
<td>A00</td>
<td>USIBMSTODB21</td>
</tr>
<tr>
<td>D01</td>
<td>DEVELOPMENT CENTER</td>
<td></td>
<td>A00</td>
<td>USIBMSTODB21</td>
</tr>
</tbody>
</table>

**Example 2: Replacing data in a given partition**

The following control statement specifies that data from the data set that is defined by the SYSREC DD statement is to be loaded into the first partition of table DSN8810.DEPT. The default input data set is SYSREC. The REPLACE option indicates that the input data is to replace only the specified partition. If the REPLACE option was specified before the PART option, REPLACE would indicate that entire table space is to be replaced, and the data is to be loaded into the specified partition. Note that the keyword DATA does not need to be specified.

```
LOAD INTO TABLE DSN8B10.DEPT PART 1 REPLACE
```

**Example 3: Loading selected records into multiple tables**

The control statement in specifies that the LOAD utility is to load certain data from the EMPLDS input data set into tables DSN8B10.EMP, SMITH.EMPEMPL, and DSN8810.DEPT. The input data set is identified by the INDDN option. The WHEN clauses indicate which records are to be loaded into each table. For the EMP and DEPT tables, the utility is to load only records that begin with the string LKA. For the EMPEMPL table, the utility is to load only records that begin with the string ABC. The RESUME YES option indicates that the table space does not need to be empty for the LOAD job to proceed. The new rows are added to the end of the tables. This example assumes that the first two tables being loaded have exactly the same format, and that the input data matches that format; therefore, no field
specifications are needed for those two INTO TABLE clauses. The third table has a
different format, so field specifications are required and are supplied in the
example.

The three tables being loaded each contain a different number of records. To
improve the sizing of the sort work data sets that the LOAD utility requires, the
number of records being loaded into each table is specified on the NUMRECS
keyword for each table.

The POSITION clauses specify the location of the fields in the input data for the
DEPT table. For each source record that is to be loaded into the DEPT table:
  • The characters in positions 7 - 9 are loaded into the DEPTNO column.
  • The characters in positions 10 - 35 are loaded into the DEPTNAME column.
  • The characters in positions 36 - 41 are loaded into the MGRNO column.
  • The characters in positions 42 - 44 are loaded into the ADMRDEPT column.

```
LOAD DATA INDDN EMPLDS
RESUME YES
INTO TABLE DSN8B10.EMP
NUMRECS 100000
WHEN (1:3)= 'LKA'
INTO TABLE SMITH.EMPEMPL
NUMRECS 100
WHEN (1:3)= 'ABC'
INTO TABLE DSN8B10.DEPT
NUMRECS 500
WHEN (1:3)= 'LKA'
(DEPTNO POSITION (7:9) CHAR,
  DEPTNAME POSITION (10:35) CHAR,
  MGRNO POSITION (36:41) CHAR,
  ADMRDEPT POSITION (42:44) CHAR)
```

*Figure 26. Example LOAD statement that loads selected records into multiple tables*

**Example 4: Loading data of different data types**

The control statement specifies that LOAD is to load data from the SYSRECPJ
input data set into table DSN8B10.PROJ. The input data set is identified by the
INDDN option. Assume that the table space that contains table DSN8B10.PROJ is
currently empty.

For each input record, data is loaded into the specified columns (that is, PROJNO,
PROJNAME, DEPTNO, and so on) to form a table row. Any other PROJ columns
that are not specified in the LOAD control statement are set to the default value.

The POSITION clauses define the starting positions of the fields in the input data
set. The ending positions of the fields in the input data set are implicitly defined
either by the length specification of the data type (CHAR length) or the length
specification of the external numeric data type (LENGTH).

The numeric data that is represented in SQL constant format (EXTERNAL format)
is converted to the correct internal format by the LOAD process and placed in the
indicated column names. The two dates (PRSTDATE and PRENDATE) are
assumed to be represented by eight digits and two separator characters, as in the
USA format (for example, 11/15/2006). The length of the date fields is given as 10
explicitly, although in many cases, the default is the same value.
Example 5: Loading data in delimited file format

The control statement specifies that data in delimited format is to be loaded into the specified columns (FILENO, DATE1, TIME1, and TIMESTMP) in table TBQB0103. The FORMAT DELIMITED option indicates that the data is in delimited format. The data is to be loaded from the SYSREC data set, which is the default.

The COLDEL option indicates that the column delimiter is a comma (,). The CHARDEL option indicates that the character string delimiter is a double quotation mark ("). The DECPt option indicates that the decimal point character is a period (.). You are not required to explicitly specify these particular characters, because they are all defaults.

```/*
//STEP3 EXEC DSNUPROC,UID='JUQBU101.LOAD2',TIME=1440,
// UTPROC='',
// SYSTEM='SSTR'
//SYSERR DD DSN=JUQBU101.LOAD2.STEP3.SYSERR,
// DISP=(MOD,DELETE,CATLG),UNIT=SYSDA,
// SPACE=(4096,(20,20),,,ROUND)
//SYSDISC DD DSN=JUQBU101.LOAD2.STEP3.SYSDISC,
// DISP=(MOD,DELETE,CATLG),UNIT=SYSDA,
// SPACE=(4096,(20,20),,,ROUND)
//SYSMAP DD DSN=JUQBU101.LOAD2.STEP3.SYSMAP,
// DISP=(MOD,DELETE,CATLG),UNIT=SYSDA,
// SPACE=(4096,(20,20),,,ROUND)
//SYSUT1 DD DSN=JUQBU101.LOAD2.STEP3.SYSUT1,
// DISP=(MOD,DELETE,CATLG),UNIT=SYSDA,
// SPACE=(4096,(20,20),,,ROUND)
//UTPRINT DD SYSOUT='',
//SORTOUT DD DSN=JUQBU101.LOAD2.STEP3.SORTOUT,
// DISP=(MOD,DELETE,CATLG),UNIT=SYSDA,
// SPACE=(4096,(20,20),,,ROUND)
//SYSSIN DD *
LOAD DATA
FORMAT DELIMITED COLDEL ',', CHARDEL '"' DECPt '.'
INTO TABLE TBQB0103
(FILENO CHAR,
 DATE1 DATE EXTERNAL,
 TIME1 TIME EXTERNAL,
 TIMESTMP TIMESTAMP EXTERNAL)
/*
//SYSREC DD *
"001", 2000-02-16, 00.00.00, 2000-02-16-00.00.00.0000
"002", 2001-04-17, 06.30.00, 2001-04-17-06.30.00.2000
"005", 2000-12-20, 24.00.00, 2000-12-20-24.00.00.0000
*/
```

Figure 27. Example of loading data of different data types

Chapter 19. LOAD
Example 6: Concatenating multiple input records

The control statement specifies that data from the SYSRECOV input data set is to be loaded into table DSN8B10.TOPTVAL. The input data set is identified by the INDDN option. The table space that contains the TOPTVAL table is currently empty.

Some of the data that is to be loaded into a single row spans more than one input record. In this situation, an X in column 72 indicates that the input record contains fields that are to be loaded into the same row as the fields in the next input record. In the LOAD control statement, CONTINUEIF(72:72)='X' indicates that LOAD is to concatenate any input records that have an X in column 72 with the next record before loading the data.

For each assembled input record (that is, after the concatenation), fields are loaded into the DSN8B10.TOPTVAL table columns (that is, MAJSYS, ACTION, OBJECT ..., DSPINDEX) to form a table row. Any columns that are not specified in the LOAD control statement are set to the default value.

The POSITION clauses define the starting positions of the fields in the assembled input records. Starting positions are numbered from the first column of the internally assembled input record, not from the start of the input records in the sequential data set. The ending positions of the fields are implicitly defined by the length specification of the data type (CHAR length).

No conversions are required to load the input character strings into their designated columns, which are also defined to be fixed-length character strings. However, because columns INFOTXT, HELPTXT, and PFKTXT are defined as 79 characters in length and the strings that are being loaded are 71 characters in length, those strings are padded with blanks as they are loaded.

```
LOAD DATA INDDN(SYSRECOV) CONTINUEIF(72:72)='X'
   INTO TABLE DSN8B10.TOPTVAL
       (MAJSYS POSITION (2) CHAR(1),
        ACTION POSITION (4) CHAR(1),
        OBJECT POSITION (6) CHAR(2),
        SRCRIT POSITION (9) CHAR(2),
        SCRTYPE POSITION (12) CHAR(1),
        HEADTXT POSITION (80) CHAR(50),
        SELTXT POSITION (159) CHAR(50),
        INFOTXT POSITION (238) CHAR(71),
        HELPTXT POSITION (317) CHAR(71),
        PFKTXT POSITION (396) CHAR(71),
        DSPINDEX POSITION (475) CHAR(2))
```

Figure 28. Example of concatenating multiple input records before loading the data

Example 7: Loading null values

The control statement specifies that data from the SYSRECST data set is to be loaded into the specified columns in table SYSIBM.SYSSTRINGS. The input data set is identified by the INDDN option. The NULLIF option for the ERRORBYTE and SUBBYTE columns specifies that if the input field contains a blank, LOAD is to place a null value in the indicated column for that particular row. The DEFAULTIF option for the TRANSTAB column indicates that the utility is to load the default value for this column if the input field value is GG. The CONTINUEIF option indicates that LOAD is to concatenate any input records that have an X in column 80 with the next record before loading the data.
Example 8: Enforcing referential constraints when loading data

The control statement specifies that data from the SYSREC input data set is to be loaded into table DSN8B10.PROJ. The default input data set is SYSREC. The table space that contains the PROJ table is not empty. RESUME YES indicates that the records are to be added to the end of the table.

The ENFORCE CONSTRAINTS option indicates that LOAD is to enforce referential constraints on the data that is being added. This option is also the default. All violations are reported in the output. All records causing these violations are not loaded and placed in the SYSDISC data set, which is the default data set for discarded records.

The CONTINUEIF option indicates that before loading the data LOAD is to concatenate any input records that have an X in column 72 with the next record.

Example 9: Loading data without enforcing referential constraints

The control statement specifies that data from the SYSRECA input data set is to be loaded into table DSN8810.ACT. The INDDN option identifies the input data set.

ENFORCE NO indicates that the LOAD utility is not to enforce referential constraints and places the table in CHECK-pending status. Use this option if you are loading data into several tables that are related in such a way that the referential constraints cannot be checked until all tables are loaded. For example, a column in table A depends on a column in table B; a column in table B depends on a column in table C; and a column in table C depends on a column in table A.

The POSITION clauses define the starting positions of the fields in the input data set. The ending positions of the fields in the input data set are implicitly defined by the length specification of the data type (CHAR length). In this case, the
characters in positions 1 through 3 are loaded into the ACTNO column, the characters in positions 5 through 10 are loaded into the ACTKWD column, and the characters in position 13 onward are loaded into the ACTDESC column. Because the ACTDESC column is of type VARCHAR, the input data needs to contain a 2-byte binary field that contains the length of the character field. This binary field begins at position 13.

```sql
//STEP1 EXEC DSNUPROC,UID='IUIQU2UB.LOAD',
// UTPROC='',
// SYSTEM='DSN'
//SYSRECAC DD DSN=IUIQU2UB.LOAD.DATA,DISP=SHR, VOL=SER=SCR03,
// UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSUT1 DD DSN=IUIQU2UB.LOAD.STEP1.SYSUT1,
// DISP=(MOD,DELETE,CATLG),
// UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SORTOUT DD DSN=IUIQU2UB.LOAD.STEP1.SORTOUT,
// DISP=(MOD,DELETE,CATLG),
// UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSIN DD *
LOAD DATA INDDN(SYSRECAC) RESUME YES
INTO TABLE DSN8B10.ACT
  (ACTNO  POSITION(1) INTEGER EXTERNAL(3),
   ACTKWD POSITION(5) CHAR(6),
   ACTDESC POSITION(13) VARCHAR)
ENFORCE NO
//*
```

**Figure 31. Example of loading data without enforcing referential constraints**

### Example 10: Loading data by using a parallel index build

The control statement specifies that data from the SYSREC input data set is to be loaded into table DSN8810.DEPT. Assume that 22,000 rows need to be loaded into table DSN8810.DEPT, which has three indexes. In this example, the SORTKEYS option is used to improve performance by forcing a parallel index build. The SORTKEYS option specifies 66,000 as an estimate of the number keys to sort in parallel during the SORTBLD phase. (This estimate was computed by using the calculation that is described in "Improving LOAD performance" on page 336.)

Because more than one index needs to be built, LOAD builds the indexes in parallel.

The SORTDEVT and SORTNUM keywords specify that the sort program is to dynamically allocate the required data sets. If sufficient virtual storage resources are available, one utility subtask pair is started to build each index. This example does not require UTPRINn DD statements because it uses DSNUPROC to invoke utility processing, which includes a DD statement that allocates UTPRINT to SYSOUT.

The CONTINUEIF option indicates that, before loading the data, LOAD is to concatenate any input records that have a plus sign (+) in column 79 and a plus sign (+) in column 80 with the next record.
Example 11: Creating inline copies using the REPLACE option

The control statement specifies that the LOAD utility is to load data from the SYSREC data set into the specified columns of table ADMF001.TB0S3902.

COPYDDN(COPYT1) indicates that LOAD is to create inline copies and write the primary image copy to the data set that is defined by the COPYT1 template. This template is defined in one of the preceding TEMPLATE control statements.

To create an inline copy, you can specify the REPLACE option, which indicates that any data in the table space is to be replaced.

CONTINUEIF(79:80)=’++’ indicates that, before loading the data, LOAD is to concatenate any input records that have a plus sign (+) in column 79 and a plus sign (+) in column 80 with the next record.

The ERRDDN(ERRDDN) and MAPDDN(MAP) options indicate that information about errors is to be written to the data sets that are defined by the ERRDDN and MAP templates. DISC ARDDN(DISCARD) specifies that discarded records (those that violate referential constraints) are to be written to the data set that is defined by the DISCARD template. WORKDDN(UT1,OUT) specifies the temporary work files for sort input and output; LOAD is to use the data set that is defined by the UT1 template for sort input and the data set that is defined by the OUT template for sort output.
Copying an inline image copy for a table space partition requires specifying the RESUME YES option with the INTO TABLE PART option and the RESUME YES SHRLEVEL NONE option. Specifying RESUME YES with the COPYDDN or RECOVERYDDN option is required for each table space partition in the LOAD utility statement.

COPYDDN(SCPY) indicates that LOAD is to create inline copies and write the primary image copy to the data set that is defined by the SCPY template. This...

Example 12: Creating inline copies using the RESUME YES option with SHRLEVEL NONE option

Creating an inline image copy for a table space partition requires specifying the RESUME YES option with the INTO TABLE PART option and the RESUME YES SHRLEVEL NONE option. Specifying RESUME YES with the COPYDDN or RECOVERYDDN option is required for each table space partition in the LOAD utility statement.

COPYDDN(SCPY) indicates that LOAD is to create inline copies and write the primary image copy to the data set that is defined by the SCPY template. This...
Example 13: Collecting statistics

This example is similar to the previous example, except that the STATISTICS option and other related options have been added so that during the LOAD job, Db2 also gathers statistics for the table space. Gathering these statistics eliminates the need to run the RUNSTATS utility after completing the LOAD operation.

The TABLE, COLUMN, and INDEX options specify that information is to be gathered for columns QT_INV_TRANSACTION, NO_DEPT, NO_PART_PREFIX, DT_TRANS_EFFECTIVE and index ID053902 for table TB053902. SAMPLE 53 indicates that LOAD is to sample 53\% of the rows when gathering statistics on non-leading-indexed columns of an index or non-indexed columns. For the index, statistics on all of the distinct values in all of the key column combinations are collected by default. FREQVAL NUMCOLS 4 COUNT 20 indicates that 20 frequent values are to be collected on the concatenation of the first four key columns.

REPORT YES indicates that the statistics are to be sent to SYSPRINT as output. UPDATE ALL and HISTORY ALL indicate that all collected statistics are to be updated in the catalog and catalog history tables.

```
//STEP1 EXEC DSNUPROC,UID='JUOSU339.LOAD1',TIME=1440,
// UTPROC='',
// SYSTEM='SSTR'
//SYSREC DD DSN=CUST.FM.CINT135.DATA,DISP=SHR,Vol=SER=FORDMD,
// UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSIN DD *
　TEMPLATE ERDDN UNIT(SYSDA)
　DSN(JUOSU339.T&TI..&ST..ERDDN)
　SPACE(50,10) TRK
　TEMPLATE UT1 UNIT(SYSDA)
　DSN(JUOSU339.T&TI..&ST..SYSUT1)
　SPACE(50,10) TRK
　TEMPLATE OUT UNIT(SYSDA)
　DSN(JUOSU339.T&TI..&ST..SYSOUT)
　SPACE(50,10) TRK
　TEMPLATE MAP UNIT(SYSDA)
　DSN(JUOSU339.T&TI..&ST..SYSMAP)
　SPACE(50,10) TRK
　TEMPLATE DISCARD UNIT(SYSDA)
　DSN(JUOSU339.T&TI..&ST..DISCARD)
　SPACE(50,10) TRK
```

Figure 34. Example of creating inline copies using the RESUME YES option with SHRLEVEL NONE option
Example 14: Loading Unicode data

The following control statement specifies that Unicode data from the REC1 input data set is to be loaded into table ADMF001.TBMG0301. The UNICODE option specifies the type of input data. Only data that satisfies the condition that is specified in the WHEN clause is to be loaded. The CCSID option specifies the three coded character set identifiers for the input file: one for SBCS data, one for mixed data, and one for DBCS data. LOG YES indicates that logging is to occur during the LOAD job.

```sql
LOAD DATA INDDN REC1 LOG YES REPLACE
  UNICODE CCSID(00367,01208,01200)
  INTO TABLE "ADMF001"."TBMG0301"
  WHEN(00004:00005 = 'X'0003')
```

Example 15: Loading data from multiple input data sets by using partition parallelism

The LOAD control statement in this example contains a series of INTO TABLE statements that specify which data is to be loaded into which partitions of table DBA01.TBLX3303. For each INTO TABLE statement:

- Data is to be loaded into the partition that is identified by the PART option. For example, the first INTO TABLE statement specifies that data is to be loaded into the first partition of table DBA01.TBLX3303.
• Data is to be loaded from the data set that is identified by the INDDN option. For example, the data from the PART1 data set is to be loaded into the first partition.

• Any discarded rows are to be written to the data set that is specified by the DISCARDDN option. For example, rows that are discarded during the loading of data from the PART1 data set are written to the DISC1 data set.

• The data is loaded into the specified columns (EMPNO, LASTNAME, and SALARY).

LOAD uses partition parallelism to load the data into these partitions.

The TEMPLATE utility control statement defines the data set naming convention for the data set that is to be dynamically allocated during the following LOAD job. The name of the template is ERR3. The ERRDDN option in the LOAD statement specifies that any errors are to be written to the data set that is defined by this ERR3 template.

```
TEMPLATE ERR3
  DSN &UT..&JO..&ST..ERR3&MO.&DAY.
  UNIT SYSDA  DISP(NEW,CATLG,CATLG)
LOAD DATA
  REPLACE
  ERRDDN ERR3
  INTO TABLE DBA01.TBLX3303
  PART 1
  INDDN PART1
  DISCARDDN DISC1
    (EMPNO  POSITION(1)  CHAR(6),
     LASTNAME  POSITION(8)  VARCHAR(15),
     SALARY  POSITION(25)  DECIMAL(9,2))
    .
    .
    .
  INTO TABLE DBA01.TBLX3303
  PART 5
  INDDN PART5
  DISCARDDN DISC5
    (EMPNO  POSITION(1)  CHAR(6),
     LASTNAME  POSITION(8)  VARCHAR(15),
     SALARY  POSITION(25)  DECIMAL(9,2))
/*
```

Figure 35. Example of loading data from individual data sets

**Example 16: Loading data from another table in the same system by using a declared cursor**

The following LOAD control statement specifies that all rows that are identified by cursor C1 are to be loaded into table MYEMP. The INCURSOR option is used to specify cursor C1, which is defined in the EXEC SQL utility control statement. Cursor C1 points to the rows that are returned by executing the statement SELECT * FROM DSN8810.EMP. In this example, the column names in table DSN8810.EMP are the same as the column names in table MYEMP. Note that the cursor cannot be defined on the same table into which Db2 is to load the data.

```
EXEC SQL
  DECLARE C1 CURSOR FOR SELECT * FROM DSN8810.EMP
ENDEXEC
LOAD DATA
```
**Example 17: Loading data partitions in parallel from a remote site by using a declared cursor**

The LOAD control statement in this example specifies that for each specified partition of table MYEMPP, the rows that are identified by the specified cursor are to be loaded. In each INTO TABLE statement, the PART option specifies the partition number, and the INCURSOR option specifies the cursor. For example, the rows that are identified by cursor C1 are to be loaded into the first partition. The data for each partition is loaded in parallel.

Each cursor is defined in a separate EXEC SQL utility control statement and points to the rows that are returned by executing the specified SELECT statement. These SELECT statement are being executed on a table at a remote server, so the three-part name is used to identify the table. In this example, the column names in table CHICAGO.DSN8810.EMP are the same as the column names in table MYEMPP.

The four partitions being loaded each contain a different number of records. To improve the sizing of the sort work data sets that the LOAD utility requires, the number of records being loaded into each partition is specified on the NUMRECS keyword for each table.

```sql
EXEC SQL
  DECLARE C1 CURSOR FOR SELECT * FROM CHICAGO.DSN8810.EMP
  WHERE EMPNO <= '099999'
ENDEXEC
EXEC SQL
  DECLARE C2 CURSOR FOR SELECT * FROM CHICAGO.DSN8810.EMP
  WHERE EMPNO > '099999' AND EMPNO <= '199999'
ENDEXEC
EXEC SQL
  DECLARE C3 CURSOR FOR SELECT * FROM CHICAGO.DSN8810.EMP
  WHERE EMPNO > '199999' AND EMPNO <= '299999'
ENDEXEC
EXEC SQL
  DECLARE C4 CURSOR FOR SELECT * FROM CHICAGO.DSN8810.EMP
  WHERE EMPNO > '299999' AND EMPNO <= '999999'
ENDEXEC
LOAD DATA
  INTO TABLE MYEMPP PART 1 REPLACE INCURSOR(C1) NUMRECS 10000
  INTO TABLE MYEMPP PART 2 REPLACE INCURSOR(C2) NUMRECS 50000
  INTO TABLE MYEMPP PART 3 REPLACE INCURSOR(C3) NUMRECS 100000
  INTO TABLE MYEMPP PART 4 REPLACE INCURSOR(C4) NUMRECS 50000
```

*Figure 36. Example of loading data partitions in parallel using a declared cursor*

**Example 18: Loading LOB data from a file**

The LOAD control statement in this example specifies that data from 000130DSN10.DSNIVPD(DSN8R130) is to be loaded into the MY_EMP_PHOTO_RESUME table. The characters in positions 1 through 6 are loaded into the EMPNO column, and the characters starting from position 7 are to be loaded into the RESUME column. CLOBF indicates that the characters in position 7 are the name of a file from which a CLOB is to be loaded.
REPLACE indicates that the new data will replace any existing data. Although no logging is to be done, as indicated by the LOG NO option, the table space is not to be set in CHECK-pending state, because NOCOPYPEND is specified.

SORTKEYS 1 indicates that one index key is to be sorted.

```sql
/******************************************************************/
/*                 LOAD LOB from file                              */
/******************************************************************/
/*LOADIT EXEC DSNUPROC,UID='LOADIT',TIME=1440,                */
/*  UTPROC='',                                             */
/*  SYSTEM='DSN',                                         */
/*SYSREC DD*                                                */
/*00010DSN10.SDSNIVPD(DSN8R130)                               */
/*SYSUT1 DD DSN=SYSADM.LOAD.SYSUT1,DISP=(MOD,DELETE,CATLG),  */
/*  UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)                   */
/*SORTOUT DD DSN=SYSADM.LOAD.SORTOUT,DISP=(MOD,DELETE,CATLG),*/
/*  UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)                   */
/*SYSIN DD *                                                 */
LOAD DATA
  REPLACE LOG NO NOCOPYPEND
  SORTKEYS 1
  INTO TABLE MY_EMP_PHOTO_RESUME
    (EMPNO  POSITION(1:6) CHAR(6),
     RESUME Position(7:31) CHAR(31) CLOBF)
```

Figure 37. Example of loading LOB data from a file

### Example 19: Loading with parallel subtasks

The following LOAD statement specifies that LOAD is to use multiple parallel subtasks, as indicated by the PARALLEL keyword. Because no value is specified with the PARALLEL keyword, Db2 determines the optimal degree of parallelism. This use of parallelism can potentially reduce the elapsed time that is required for loading large amounts of data.

```sql
LOAD DATA
  PARALLEL
  RESUME YES
  SHRLEVEL NONE
  INDDN INPUT1
  EBCDIC
  CONTINUEIF(80:80)='-'
  INTO TABLE SCTX1300.TB_HISTORY_PART
    (HISTORY_DAILY Position(1:11) INT EXTERNAL,
     HISTORY_ROWNUM Position(13:23) INT EXTERNAL,
     HISTORY_CUSTOMER_ID Position(25:35) INT EXTERNAL,
     HISTORY_CUSTOMER_ACCOUNT_ID Position(37:50) DECIMAL EXTERNAL,
     HISTORY_CUSTOMER_DISTRICT_ID Position(52:53) CHAR,
     HISTORY_CUSTOMER_WAREHOUSE_ID Position(55:64) CHAR,
     HISTORY_DISTRICT_ID Position(66:67) CHAR,
     HISTORY_TRANSACTION_ID Position(69:70) CHAR,
     HISTORY_WAREHOUSE_ID Position(72:81) CHAR,
     HISTORY_DATE Position(83:108) TIMESTAMP EXTERNAL,
     HISTORY_AMOUNT Position(110:126) DECIMAL EXTERNAL,
     HISTORY_STATUS Position(128:144) VARCHAR,
     HISTORY_DATA Position(3874:3899) VARCHAR)
```

Related reference:
- [Db2 Sort for z/OS](#)

Related information:
- [DFSORT Application Programming Guide](#)
Chapter 20. MERGECOPY

The MERGECOPY online utility merges copies or inline copies that other utilities produce. The COPY and COPYTOCOPY utilities produce image copies, and the LOAD and REORG utilities produce inline copies.

The utility can merge several incremental copies of a table space to make one incremental copy. It can also merge incremental copies with a full image copy to make a new full image copy. You cannot run MERGECOPY on concurrent copies or FlashCopy image copies.

MERGECOPY operates on the image copy data sets of a table space, and not on the table space itself.

If you are creating copies in a JES3 environment, ensure that sufficient units are available to mount the required image copies. In a JES3 environment, if the number of image copies that are to be restored exceeds the number of available online and offline units, and if the MERGECOPY job successfully allocates all available units, the job waits for more units to become available.

Output

Output from the MERGECOPY utility consists of one of the following types of copies:

- A new single incremental image copy
- A new full image copy

You can create the new image copy for the local or recovery site.

Authorization required

To execute this utility, you must use a privilege set that includes one of the following authorities:

- IMAGCOPY privilege for the database
- DBADM, DBCTRL, or DBMAINT authority for the database. If the object on which the utility operates is in an implicitly created database, DBADM authority on the implicitly created database or DSNDB04 is required.
- DATACCESS authority
- SYSCTRL or SYSADM authority

An ID with installation SYSOPR authority can also execute MERGECOPY, but only on a table space in the DSNDB01 or DSNDB06 database.

Restrictions on running MERGECOPY

- MERGECOPY cannot merge image copies into a single incremental image copy for the other site, that is:
  - At local sites, you cannot use RECOVERYDDN with NEWCOPY NO.
  - At recovery sites, you cannot use COPYDDN with NEWCOPY NO.
- When none of the keywords NEWCOPY, COPYDDN, or RECOVERYDDN is specified, the default, NEWCOPY NO COPYDDN(SYSCOPY), is valid for the local site only.
- You cannot run MERGECOPY on concurrent copies.
You cannot run the MERGECOPY utility on the DSNDB01.DBD01, DSNDB01.SYSUTILX, DSNDB06.SYSTSCPY, or DSNDB01.SYSDBDXA table spaces, because you cannot make incremental copies of those table spaces.

MERGECOPY cannot be run on a table space during the period after RECOVER is run to a point in time before materialization of pending definition changes and before REORG is run to complete the point-in-time recovery process.

**Execution phases of MERGECOPY**

The MERGECOPY utility operates in these phases:

<table>
<thead>
<tr>
<th>Phase</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTILINIT</td>
<td>Performs initialization</td>
</tr>
<tr>
<td>MERGECOPY</td>
<td>Merges incremental copies</td>
</tr>
<tr>
<td>UTILTERM</td>
<td>Performs cleanup</td>
</tr>
</tbody>
</table>

**Syntax and options of the MERGECOPY control statement**

The MERGECOPY utility control statement, with its multiple options, defines the function that the utility job performs.

You can create a control statement with the ISPF/PDF edit function. After creating it, save it in a sequential or partitioned data set. When you create the JCL for running the job, you can use the SYSIN DD statement to specify the name of the data set that contains the utility control statement.

**Syntax diagram**
Option descriptions

**LIST listdef-name**

Specifies the name of a previously defined LISTDEF list name that contains only table spaces. You can specify one LIST keyword per MERGECOPY control statement. Do not specify LIST with the TABLESPACE keyword. MERGECOPY is invoked once for each table space in the list. This utility will only process clone data if the CLONE keyword is specified. The use of CLONED YES on the LISTDEF statement is not sufficient.

The partitions or partition ranges can be specified in a list.

**TABLESPACE database-name.table-space-name**

Specifies the table space that is to be copied, and, optionally, the database to which it belongs.

- **database-name**
  - The name of the database that the table space belongs to. The default value is **DSND04**.

- **table-space-name**
  - The name of the table space whose incremental image copies are to be merged.

You cannot run the MERGECOPY utility on the DSND01.DBD01, DSND01.SYSUTILX, DSND06.SYSTSCPY, or DSND01.SYSDBDXA table spaces, because you cannot make incremental copies of those table spaces. Because MERGECOPY does not directly access the table space whose copies it is merging, it does not interfere with concurrent access to that table space.

**DSNUM**

Identifies the table space or a partition or data set within the table space that is to be merged. DSNUM is optional.

- **ALL**
  - Merges the entire table space.

- **integer**
  - Is the number of a partition or data set that is to be merged. The maximum is 4096.

  For a partitioned table space, the integer is its physical partition number.

  For a nonpartitioned table space, find the integer at the end of the data set name as cataloged in the VSAM catalog. The data set name has the following format, where $y$ is either I or J, $z$ is either 1 or 2, and $nnn$ is the data set integer:

  $\text{catname.DSNDBx.dbname.tsname.y000z.Annn}$

You cannot specify DSNUM and LIST in the same MERGECOPY control statement. Use PARTLEVEL on the LISTDEF instead. If image copies were taken by data set (rather than by table space), MERGECOPY must use the copies by data set.

**CLONE**

Indicates that MERGECOPY is to process only image copy data sets that were taken against clone objects. This utility will only process clone data if the CLONE keyword is specified. The use of CLONED YES on the LISTDEF statement is not sufficient.
WORKDDN ddname
Specifies a DD statement for a temporary data set or template, which is to be used for intermediate merged output. WORKDDN is optional.

`ddname` is the DD name. The **default** value is **SYSUT1**.

Use the WORKDDN option if you are not able to allocate enough data sets to execute MERGECOPY; in that case, a temporary data set is used to hold intermediate output. If you omit the WORKDDN option, you might find that only some of the image copy data sets are merged. When MERGECOPY has ended, a message is issued that tells the number of data sets that exist and the number of data sets that have been merged. To continue the merge, repeat MERGECOPY with a new output data set.

NEWCOPY
Specifies whether incremental image copies are to be merged with the full image copy. NEWCOPY is optional.

**NO**
Merges incremental image copies into a single incremental image copy but does not merge them with the full image copy.

**YES**
Merges all incremental image copies with the full image copy to form a new full image copy.

COPYDDN (ddname1,ddname2)
Specifies the DD statements for the output image copy data sets at the local site. `ddname1` is the primary output image copy data set. `ddname2` is the backup output image copy data set. COPYDDN is optional.

The **default** value is **COPYDDN(SYSCOPY)**, where SYSCOPY identifies the primary data set.

The COPYDDN keyword specifies either a DD name or a TEMPLATE name specification from a previous TEMPLATE control statement. If utility processing detects that the specified name is both a DD name in the current job step and a TEMPLATE name, the utility uses the DD name.

RECOVERYDDN (ddname3,ddname4)
Specifies the DD statements for the output image copy data sets at the recovery site. You can have a maximum of two output data sets; the outputs are identical. `ddname3` is the primary output image copy data set. `ddname4` is the backup output image copy data set. RECOVERYDDN is optional. No default value exists for RECOVERYDDN.

The RECOVERYDDN keyword specifies either a DD name or a TEMPLATE name specification from a previous TEMPLATE control statement. If utility processing detects that the specified name is both a DD name in the current job step and a TEMPLATE name, the utility uses the DD name.

**Related reference:**

- Chapter 34, “TEMPLATE,” on page 793
- Chapter 18, “LISTDEF,” on page 215

**Data sets that MERGECOPY uses**

The MERGECOPY utility uses a number of data sets during its operation.
The following table lists the data sets that MERGECOPY uses. The table lists the DD name that is used to identify the data set, a description of the data set, and an indication of whether it is required. Include statements in your JCL for each required data set and any optional data sets that you want to use.

**Table 52. Data sets that MERGECOPY uses**

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSIN</td>
<td>Input data set that contains the utility control statement.</td>
<td>Yes</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>Output data set for messages.</td>
<td>Yes</td>
</tr>
<tr>
<td>Image copy data set</td>
<td>Image copy data set that contains the resulting image copy. Specify its DD name with the COPYDDN option of the utility control statement. The default DD name is SYSCOPY.</td>
<td>Yes</td>
</tr>
<tr>
<td>Work data set</td>
<td>A temporary data set that is used for intermediate merged output. Specify its DD name with the WORKDDN option of the utility control statement. The default DD name is SYSUT1.</td>
<td>Yes</td>
</tr>
<tr>
<td>Input data sets</td>
<td>Image copy data sets that you can preallocate. You define the DD names.</td>
<td>No</td>
</tr>
</tbody>
</table>

**Table space**

Object whose copies are to be merged.

The following object is named in the utility control statement and does not require a DD statement in the JCL:

**Data sets**

The input data sets for the merge operation are dynamically allocated. To merge incremental copies, allocate in the JCL a work data set (WORKDDN) and up to two new copy data sets (COPYDDN) for the utility job. You can allocate the data sets to tape or disk. If you allocate them to tape, you need an additional tape drive for each data set.

With the COPYDDN option of MERGECOPY, you can specify the DD names for the output data sets. The option has the format COPYDDN (ddname1,ddname2), where ddname1 is the DD name for the primary output data set in the system that currently runs Db2, and ddname2 is the DD name for the backup output data set in the system that currently runs Db2. The default for ddname1 is SYSCOPY.

The RECOVERYDDN option of MERGECOPY lets you specify the output image copy data sets at the recovery site. The option has the format RECOVERYDDN (ddname3, ddname4), where ddname3 is the DD name for the primary output image copy data set at the recovery site, and ddname4 is the DD name for the backup output data set at the recovery site.

**Defining the work data set**

The work data set should be at least equal in size to the largest input image copy data set that is being merged. Use the same DCB attributes that are used for the image copy data sets.

**Related concepts:**
Concurrency and compatibility for MERGECOPY

The MERGECOPY utility has certain concurrency and compatibility characteristics associated with it.

Db2 treats individual data and index partitions as distinct target objects. Utilities that operate on different partitions of the same table space or index space are compatible.

The following table shows the restrictive state that the utility sets on the target object.

Table 53. Claim classes of MERGECOPY operations.

<table>
<thead>
<tr>
<th>Target</th>
<th>MERGECOPY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table space or partition</td>
<td>UTRW</td>
</tr>
</tbody>
</table>

Legend:

UTRW - Utility restrictive state - read-write access allowed.

MERGECOPY can run concurrently on the same target object with any utility except the following utilities:
- COPY TABLESPACE
- LOAD
- MERGECOPY
- MODIFY
- RECOVER
- REORG TABLESPACE
- UNLOAD (only when from the same image copy data set)

The target object can be a table space or partition.

Full or incremental image copy

When you use the MERGECOPY utility to merge copies, you can choose whether you want to create an incremental image copy or a full image copy. In general, creating a new full image copy is recommended.

Creating a full image copy with MERGECOPY has the following advantages:
- A new full image copy creates a new recovery point.
- The additional time that it takes to create a new full image copy does not have any adverse effect on the access to the table space. The only concurrency implication is the access to SYSIBM.SYSCOPY.
- The range of log records that need to be applied by the RECOVER utility is the same for both the new full image copy and the merged incremental image copy.
- If the copies are on tape, only one tape drive is required for image copies during recovery.

If you want MERGECOPY to create a new full image copy, specify NEWCOPY YES. The utility inserts an entry for the new full image copy into the SYSIBM.SYSCOPY catalog table.
Otherwise, if you do not specify NEWCOPY or specify NEWCOPY NO, MERGECOPY creates an incremental image copy. The utility deletes all SYSCOPY records of the incremental image copies that have been merged and replaces them with an entry for the new incremental image copy.

Regardless of what you specify for NEWCOPY, if any of the input data sets might not be allocated or you did not specify a temporary work data set (WORKDDN), the utility performs a partial merge.

**Recommendation:** Use MERGECOPY NEWCOPY YES immediately after each incremental image copy. When you use this option, dates become a valid criterion for deletion of image copy data sets and archive logs. A minimum number of tape drives are allocated for MERGECOPY and RECOVER execution.

If you merge an inline copy with incremental copies, the result is a full inline copy. The data set is logically equivalent to a full image copy, but the data within the data set differs in some respects.

### How MERGECOPY determines which input copy to use

The MERGECOPY utility uses as input the image copies that match the current site.

If MERGECOPY is running at the local site, the local site image copies are chosen as the input to be merged. If MERGECOPY is running at the recovery site, the recovery site image copies are chosen as the input to be merged.

MERGECOPY does not accept a FlashCopy image copy as input.

### Using MERGECOPY with individual data sets

Use MERGECOPY on copies of an entire table space, on individual data sets, or on partitions. However, MERGECOPY can only merge incremental copies of the same type. That is, you cannot merge incremental copies of an entire table space with incremental copies of individual data sets to form new incremental copies.

**About this task**

The attempt to mix the two types of incremental copies results in the following messages:

- DSNU460I DSNUBCLO - IMAGE COPIES INCONSISTENT.
  MERGECOPY REQUEST REJECTED
- DSNU010I DSNUGBAC - UTILITY EXECUTION COMPLETE,
  HIGHEST RETURN CODE=4

With the NEWCOPY YES option, however, you can merge a full image copy of a table space with incremental copies of the table space and of individual data sets to make a new full image copy of the table space.

If the image copy data sets that you want to merge reside tape, refer to “How the RECOVER utility retains tape mounts” on page 509 for general information about specifying the appropriate parameters on the DD statements.
Using MERGECOPY or COPY

COPY and MERGECOPY can create a full image copy. COPY is required after a LOAD or REORG with LOG NO unless an inline copy is created. However, in other cases an incremental image copy followed by MERGECOPY is a valid alternative.

Avoiding MERGECOPY LOG RBA inconsistencies

MERGECOPY does not use information that was logged between the time of the most recent image copy and the time when MERGECOPY was run. Therefore, you cannot safely delete all log records that were created before you ran MERGECOPY.

About this task

You can safely delete all log records if you run MODIFY RECOVERY and specify the date when MERGECOPY was run as the value of DATE.

Procedure

1. To delete all log information that is included in a copy that MERGECOPY makes:
2. Find the record of the copy in the catalog table SYSIBM.SYSCOPY by selecting database name, table space name, and date (columns DBNAME, TSNAME, and TIMESTAMP).
3. Column START_RBA contains the RBA of the last image copy that MERGECOPY used. Find the record of the image copy that has the same value of START_RBA.
4. In that record, find the date in column TIMESTAMP. You can use MODIFY RECOVERY to delete all copies and log records for the table space that were made before that date.

Results

RECOVER uses the LOG RBA of image copies to determine the starting point in the log that is needed for recovery. Normally, a timestamp directly corresponds to a LOG RBA. Because of this, and because MODIFY uses dates to clean up recovery history, you might decide to use dates to delete old archive log tapes. This decision might cause a problem if you use MERGECOPY. MERGECOPY inserts the LOG RBA of the last incremental image copy into the SYSCOPY row that is created for the new image copy. The date that is recorded in the TIMESTAMP column of SYSCOPY row is the date that MERGECOPY was executed.

Termination or restart of MERGECOPY

You can terminate and restart the MERGECOPY utility.

You can terminate the a MERGECOPY utility job using the TERM UTILITY command.

You can restart MERGECOPY but by default, MERGECOPY restarts at the beginning of the current phase. You can also restart MERGECOPY from the last commit point after receiving an out-of-space condition.

Related tasks:

- “Terminating an online utility” on page 43
Sample MERGECOPY control statements

Use the sample control statements as models for developing your own MERGECOPY control statements.

**Example 1: Creating a merged incremental copy**

The control statement in this example specifies that the MERGECOPY utility is to merge incremental image copies from table space DSN8S11C into a single incremental image copy. The NEWCOPY NO option indicates that these incremental copies are not to be merged with the full image copy. The COPYDDN option specifies that the output image copies are to be written to the data sets that are defined by the COPY1 and COPY2 DD statements.

```plaintext
//STEP1 EXEC DSNUPROC,UID='IUJMU107.MERGE1',
    // UTPROC='',SYSTEM='DSN'
//COPY1 DD DSN=IUJMU107.MERGE1.STEP1.COPY1,DISP=(MOD,CATLG,CATLG),
    // UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//COPY2 DD DSN=IUJMU107.MERGE1.STEP1.COPY2,DISP=(MOD,CATLG,CATLG),
    // UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSUT1 DD DSN=IUJMU107.MERGE1.STEP1.SYSUT1,DISP=(MOD,DELETE,CATLG),
    // UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSIN DD *
MERGECOPY TABLESPACE DSN8D11P.DSN8S11C
    COPYDDN (COPY1,COPY2)
    NEWCOPY NO
```

*Figure 38. Example of creating a merged incremental copy*

**Example 2: Creating merged incremental copies and using template switching**

Each MERGECOPY control statement in this example specifies that MERGECOPY is to merge incremental image copies from the specified table space into a single incremental image copy for that table space. For each control statement, the COPYDDN option specifies that the output image copies are to be written to data sets that are defined by the T1 template. The T1 template has specified the LIMIT option. This means that the output image copies are to be written to DASD, if the output image copy size is less than 5 MB. If the limit is exceeded, template switching from template T1 to template T5 takes place and the output image copies are to be written to TAPE. This template is defined in the TEMPLATE utility control statement.
Example 3: Creating a merged full image copy

The control statement in this example specifies that MERGECOPY is to merge all incremental image copies with the full image copy from table space DSN8S11C to create a new full image copy.

```
//STEP1 EXEC DSNUPROC,UID='JULTU224.MERGE',
// UTPROC='',
// SYSTEM='SSTR'
//SYSUT1 DD DSN=JULTU224.MERGE.STEP1.SYSUT1,DISP=(MOD,DELETE,CATLG),
// UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSIN DD *
  TEMPLATE T1 UNIT(SYSDA) SPACE CYL
    DSN(T1.&SN..T&T1..COPY&IC.&LOCREM.)
    LIMIT(5 MB,T5)
  TEMPLATE T5 UNIT(3BO)
    DSN(T5.&SN..T&T1..COPY&IC.&LOCREM.)
  MERGECOPY TABLESPACE DBLT2401.TPLT2401 DSNUM ALL NEWCOPY NO
    COPYDDN(T1)
  MERGECOPY TABLESPACE DBLT2401.TLLT24A1 DSNUM ALL NEWCOPY NO
    COPYDDN(T1)
  MERGECOPY TABLESPACE DBLT2401.TLLT24A2 DSNUM ALL NEWCOPY NO
    COPYDDN(T1)
  MERGECOPY TABLESPACE DBLT2401.TLLT24A3 DSNUM ALL NEWCOPY NO
    COPYDDN(T1)
  MERGECOPY TABLESPACE DBLT2401.TLLT24A4 DSNUM ALL NEWCOPY NO
    COPYDDN(T1)
```

Example 4: Using MERGECOPY with CLONE keyword

The following control statement specifies that MERGECOPY is to process only image copy data sets that were taken against clone objects.

```
MERGECOPY TABLESPACE DBIQU001.TPIQU001 DSNUM ALL CLONE NEWCOPY YES
    COPYDDN(COPYTB1)
```

Related reference:

Chapter 34, “TEMPLATE,” on page 793
Chapter 21. MODIFY RECOVERY

Run the MODIFY RECOVERY utility regularly to remove outdated information from SYSIBM.SYSCOPY and SYSIBM.SYSLGRNX. These tables, particularly SYSIBM.SYSLGRNX, can become very large and take up a considerable amount of space. By deleting outdated information from these tables, you can help improve the performance of processes that access data from these tables.

MODIFY RECOVERY deletes records from the following tables:
- SYSIBM.SYSCOPY catalog table
- SYSIBM.SYSOBDS catalog table
- SYSIBM.SYSLGRNX directory table
The utility also deletes entries from the DBD.

You specify the criteria for deleting these records in the MODIFY RECOVERY statement. You can delete records based on age or a specific date. Alternatively, you can specify that you want to keep a specified number of the most recent records.

MODIFY RECOVERY can delete records for an entire table space, partition, or data set. For all indexes that were defined with the COPY YES attribute on the target table space, the utility automatically deletes the SYSCOPY and SYSLGRNX records that meet the specified criteria.

You can also use MODIFY RECOVERY to recycle Db2 version numbers for reuse.

Restriction: If a table space is in REORG-pending (REORP) status because a RECOVER job was run to recover the data to a point in time before the materialization of pending definition changes, you cannot run MODIFY RECOVERY on that table space. You must run a REORG TABLESPACE job to complete the point-in-time recovery process before you run MODIFY RECOVERY.

SYSIBM.SYSCOPY and SYSIBM.SYSLGRNX do not contain records for DSNDB06.SYSTSCPY, DSNDB01.SYSUTILX, DSNDB01.DBDD01, or DSNDB01.SYSDBDXA. If you run MODIFY RECOVERY on these table spaces, no SYSCOPY or SYSLGRNX records are deleted.

Output

In addition to deleting catalog and directory records and entries from the DBD, MODIFY RECOVERY can also affect the following items:

SYSIBM.SYSCOPY records
In most cases, MODIFY RECOVERY inserts a row into SYSIBM.SYSCOPY to record the RBA or LRSN of the most recently deleted SYSCOPY or SYSLGRNX record. That row has ICTYPE='M' and STYPE='R'.

MODIFY RECOVERY does not insert this row in SYSCOPY when no SYSLGRNX rows are deleted or when the only SYSCOPY rows that were deleted are rows with the following values:
- ICTYPE='F' and STYPE is blank (full image copy)
- ICTYPE='F' and STYPE='C' (concurrent copy of the "I" instance)
• ICTYPE='F' and STYPE='J' (concurrent copy of the 'J' instance)
• ICTYPE='I' (incremental image copy)
• ICTYPE='Q' (quiesce point)

COPY-pending status

If MODIFY RECOVERY deletes at least one SYSCOPY record, and the target table space or partition is not recoverable from remaining SYSCOPY records or from system-level backups, the target object is placed in COPY-pending status.

Version numbers

For table spaces and indexes that are defined with COPY YES, MODIFY RECOVERY updates the OLDEST_VERSION column of the following catalog tables:
• SYSIBM.SYSTABLESPACE
• SYSIBM.SYSTABLEPART
• SYSIBM.SYSINDEXES
• SYSIBM.SYSINDEXPART

When MODIFY RECOVERY deletes all of the SYSCOPY records that contain an OLDEST_VERSION value of 0 for a table space, MODIFY RECOVERY deletes the corresponding rows for that table space from SYSIBM.SYSOBDS. The reason is because point-in-time recovery for the table space is no longer possible.

Authorization required

To execute this utility, you must use a privilege set that includes one of the following authorities:
• IMAGCOPY privilege for the database
• System DBADM authority
• DBADM authority
• SYSCTRL or SYSADM authority
• DBADM, DBCTRL, or DBMAINT authority for the database

If the object on which the utility operates is in an implicitly created database, DBADM authority on the implicitly created database or DSNDB04 is required.

An ID with installation SYSOPR authority can also execute MODIFY RECOVERY, but only on a table space in the DSNDB01 or DSNDB06 database.

Execution phases of MODIFY RECOVERY

The MODIFY RECOVERY utility operates in these phases:

UTILINIT
  Initialization and setup.

MODIFY
  Records are deleted.

UTILTERM
  Cleanup.
Option descriptions

**LIST listdef-name**
Specifies the name of a LISTDEF list. This list must contain only table spaces or table space partitions. MODIFY RECOVERY processes each object in the list.

Related information:
"Referencing LISTDEF lists in other utility jobs" on page 231

**TABLESPACE**
Specifies the table space for which records are to be deleted.

\[\text{database-name} . \text{table-space-name}\]
The name of the database and table space. The default value for \text{database-name} is **DSNDB04**.

**DSNUM integer**
Identifies a single partition or data set for which records are to be deleted.

\[\text{integer}\]
The number of the partition or data set.

For a partitioned table space, specify the physical partition number. The maximum is 4096.

For a nonpartitioned table space, use the data set integer at the end of the data set name as cataloged in the VSAM catalog. The data set name has the following format:

\[\text{catname} . \text{DSNDBx.dbname.tsname.yznnn.Annn}\]

where \(y\) is either I or J, \(z\) is either a 1 or 2, and \(nnn\) is the data set integer.

If you specify DSNUM \(\text{integer}\), MODIFY RECOVERY does not delete any partition SYSCOPY records that have an RBA that is greater than that of the earliest point to which the entire table space could be recovered. (That point might indicate a full image copy, a LOAD operation with LOG YES or a REORG operation with LOG YES.)
If the partition that you specify is in PRO restricted status, the RETAIN value is set to LAST(2).

If you specify DSNUM integer for a partitioned table space, MODIFY RECOVER also deletes SYSCOPY records for all partitioned index spaces and updates the version numbers in the SYSBM.SYSINDEXES catalog table. Db2 does not perform these functions for the nonpartitioned indexes.

**ALL**
Records are deleted for the entire data set and table space. The default value is **ALL**.

If image copies are taken by partition or data set and you specify DSNUM ALL, the table space is placed in COPY-pending status if a full image copy of the entire table space does not exist.

If DSNUM ALL is implicitly or explicitly specified for a table space that has a partition in PRO restricted status, MODIFY RECOVERY fails.

**CLONE**
Indicates that MODIFY RECOVERY is to delete SYSCOPY and SYSLGRNX records for only clone objects. If CLONE is not specified, only records for the base objects are deleted.

If you want MODIFY RECOVERY to process clone objects in a LISTDEF list, specify CLONE in the MODIFY RECOVERY statement. (The CLONED YES option in the LISTDEF statement only specifies whether to include clone objects in list; it has no effect on whether MODIFY RECOVERY processes those objects.)

**DELETE**
Indicates that records are to be deleted.

**AGE integer**
Deletes all SYSCOPY and SYSLGRNX records that are older than the specified number of days.

`integer`
The number of days. The value can range from 0 to 32767. Records that are created today are of age 0 and cannot be deleted by this option.

`(*)`
Deletes all records, regardless of their age.

**DATE integer**
Deletes all SYSCOPY and SYSLGRNX records that were written before the specified date.

`integer`
The date. Specify this value in eight- or six-character format: `yyyyymmdd` or `yymmdd`, where `yyyy` or `yy` is the year, `mm` is the month, and `dd` is the day.

If you specify a six-character date, Db2 checks the system clock and converts the date to the most recent, previous eight-character equivalent.

`(*)`
Deletes all records, regardless of the date on which they were written.
environment, if no SYSLGRNX records were created in DB2 9 new-function
mode or later and no SYSCOPY records are deleted, MODIFY RECOVERY
does not deletes SYSLGRNX records, even if they fit the deletion criteria.

**RETAIN**
Indicates that certain records are to be retained. Older records are deleted.

**LAST** *(integer)*
Specifies the number of recent records to retain in SYSIBM.SYSCOPY.

**LOGLIMIT**
Deletes records that are older than the oldest archive log timestamp. For
data sharing, this value is the oldest archive log timestamp across all the
members.

The utility determines the oldest archive log timestamp by querying the
BDS. If the BDS is not available for one of the members and the
corresponding member is quiesced, this BDS is ignored.

If you use a replication product that reads Db2 log records, consider using
the RETAIN LOGLIMIT option to ensure that the version information for
table spaces and indexes is retained with the same duration as the logs.

**GDGLIMIT**
If the most recent record in SYSIBM.SYSCOPY refers to a generation data
set (GDS), GDGLIMIT specifies that the utility is to consider only those
records that reference the same GDG and retain as many as it can without
exceeding the GDG limit value.

Db2 does not consider other GDGs that are referenced by
SYSIBM.SYSCOPY records. The records that reference other GDGs are
deleted in accordance with the deletion date.

**LAST** *(integer)*
If the most recent record in SYSIBM.SYSCOPY does not refer to a GDS,
LAST specifies the number of recent records to retain in
SYSIBM.SYSCOPY.

**LOGLIMIT**
If the most recent record in SYSIBM.SYSCOPY does not refer to a GDS,
LOGLIMIT deletes records that are older than the oldest archive log
timestamp.

When you specify RETAIN, the utility determines a cleanup date by checking
SYSCOPY records. The utility considers only local primary full image copies
(CTYPE=F and ICBACKUP=blank) with the specified DSNUM value. The
utility determines a date, not a complete timestamp. As a result, more copies
might be kept than are specified by RETAIN. For example, if you specify
RETAIN LAST (2) and the most recent five copies have been taken on the same
day, all five copies remain in SYSCOPY.

**NOCOPYPEND**
Specifies that the table space is not to be placed in COPY-pending status if
MODIFY RECOVERY deletes all image copy records from SYSIBM.SYSCOPY.

NOCOPYPEND does not remove any COPY-pending status that was set prior
to the MODIFY RECOVERY utility being run. NOCOPYPEND also does not
set informational COPY-pending (ICOPY) status for indexes.
Specify NOCOPYPEND only if the data in the table space can be easily re-created if the data is lost. If an image copy is not taken after MODIFY RECOVERY is run, the table space cannot be recovered by using the RECOVER utility and data might be lost.

Before running MODIFY RECOVERY

Before you run MODIFY RECOVERY, complete the following actions:

- **Make sure that DSNDDB01.SYSLGRNX is not in a restrictive state.** Because MODIFY RECOVERY updates DSNDDB01.SYSLGRNX, any restrictive status on this table space might cause the utility to abend.

  A prior MODIFY RECOVERY run on DSNDDB01.SYSLGRNX could inadvertently lead to this situation. For example, if all recovery information was deleted by the specified age or date criteria, DSNDDB01.SYSLGRNX is set to COPY-pending status.

- **Reset RECOVER-pending status for any table spaces on which you plan to run MODIFY RECOVERY.** You cannot run MODIFY RECOVERY on a table space that is in RECOVER-pending status.

**Recommendation:** If you plan to use MODIFY RECOVERY to delete SYSCOPY records, first run the REPORT utility with the RECOVERY option. This utility reports all SYSCOPY records for the object at the specified site. Looking at this report first helps you avoid deleting the wrong records.

**Recommendation:** To improve the performance of MODIFY RECOVERY and reduce contention on SYSLGRNX, run the REORG TABLESPACE utility on DSNDDB01.SYSLGRNX on a regular basis.

Related information:
- Appendix B, “Advisory or restrictive states,” on page 1037
- “Resetting RECOVER-pending or REBUILD-pending status” on page 507
- “Syntax and options of the REPORT control statement” on page 704
- “REPORT output” on page 710
- “Syntax and options of the REORG TABLESPACE control statement” on page 556

Data sets that MODIFY RECOVERY uses

MODIFY RECOVERY always requires the following two data sets:

**Table 54. Data sets that MODIFY RECOVERY uses**

<table>
<thead>
<tr>
<th>Data set</th>
<th>Default DD name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-stream data</td>
<td>SYSIN</td>
<td>An input data stream that contains the utility control statement.</td>
</tr>
<tr>
<td>Output data set</td>
<td>SYSPRINT</td>
<td>An output data set for utility messages.</td>
</tr>
</tbody>
</table>
Concurrency and compatibility for MODIFY RECOVERY

During processing, MODIFY RECOVERY sets the claim class of the target table space or partition to UTRW (Utility restrictive state - Read-write access allowed).

MODIFY RECOVERY can run concurrently on the same target object with any utility except the following utilities:
• COPY TABLESPACE
• LOAD
• MERGECOPY
• MODIFY RECOVERY
• RECOVER TABLESPACE
• REORG TABLESPACE

Db2 treats individual data partitions as distinct target objects. Utilities that operate on different partitions of the same table space are compatible.

Termination or restart of MODIFY RECOVERY

You can use the TERM UTILITY command to terminate MODIFY RECOVERY in any phase without any integrity exposure.

You can restart a MODIFY RECOVERY utility job, but it starts from the beginning again.

Related information:
“Terminating an online utility” on page 43
“Restarting an online utility” on page 44

How MODIFY RECOVERY deletes rows

You can use the MODIFY RECOVERY utility to delete specific rows from Db2 catalog and directory tables in certain conditions.

Deletion of SYSLGRNX and SYSCOPY rows for a single partition or the entire table space

You can use the MODIFY RECOVERY utility to delete rows from the SYSIBM.SYSLGRNX directory table and SYSIBM.SYSCOPY catalog table. Use the DSNUM option to specify whether to delete rows for a single partition or for the entire table space. The DSNUM value that you specify (ALL or integer) depends on the type of image copies that exist for the table space.

Use the following guidelines to determine whether to use DSNUM ALL or DSNUM integer:
• If image copies exist at only the partition level, use DSNUM integer.
• If image copies exist at only the data set level for a nonpartitioned table space, use DSNUM ALL. If DSNUM integer is used, SYSLGRNX records are not deleted.
• If image copies exist at only the table space or index space level, use DSNUM ALL.
• If image copies exist at both the partition level and the table space or index space level, use DSNUM ALL.
**Restriction:** In this case, if you use DSNUM integer, MODIFY RECOVERY does not delete any SYSCOPY or SYSLGRNX records that are newer than the oldest recoverable point at the table space or index space level.

- If image copies exist at both the data set level and the table space level for a nonpartitioned table space, use DSNUM ALL.

**Restriction:** In this case, if you use DSNUM integer, MODIFY RECOVERY does not delete any SYSCOPY or SYSLGRNX records that are newer than the oldest recoverable point at the table space level.

- If image copies exist at the table space level and the table space is subsequently converted from a nonpartitioned table space to a partitioned table space, use DSNUM ALL. For example, you would use DSNUM ALL if image copies exist on a simple table space or on a single-table segmented (non-universal) table space, but that table space is subsequently converted to a partition-by-growth universal table space.

**Restriction:** In this case, if you use DSNUM integer, MODIFY RECOVERY does not delete any SYSCOPY or SYSLGRNX records that are newer than the oldest recoverable point before table space conversion.

The preceding guidelines pertain to all image copies, regardless of how they were created, including those copies that were created by COPY, COPYTOCOPY, LOAD, REORG TABLESPACE, or MERGECOPY.

If MODIFY RECOVERY deletes SYSCOPY or SYSLGRNX rows, it inserts a row into SYSCOPY with the following values:

- ICTYPE='M'
- STYPE='R'
- A START_RBA value that is equal to the START_RBA value of the SYSCOPY or SYSLGRNX row that was most recently deleted.

However, suppose that MODIFY RECOVERY deletes SYSCOPY rows with an ICTYPE value of 'C', 'I' or 'Q' but does not delete any SYSLGRNX rows. In this case, MODIFY RECOVERY does not insert rows into SYSCOPY with the values ICTYPE='M', STYPE='R'.

**Deletion of SYSLGRNX rows when no SYSCOPY rows exist**

Use the AGE or DATE options when you want to delete SYSLGRNX rows and no SYSCOPY rows meet the deletion criteria. The SYSLGRNX rows are deleted based on the AGE or DATE specified. The RECOVER utility uses this information to determine whether it has all of the necessary information for the recovery of objects.

**Deletion of recovery rows for indexes**

When MODIFY RECOVERY processes a table space, the utility deletes SYSCOPY and SYSLGRNX rows that meet the AGE and DATE criteria for related indexes with the COPY YES attribute.

**Deletion of all image copy entries**

You can use MODIFY RECOVERY to delete all image copy entries for a table space or data set. In this case, MODIFY RECOVERY places the object in COPY-pending (COPY) restrictive status and issues message DSNU572I.
Deletion of SYSOBDS entries

MODIFY RECOVERY removes entries that the database manager inserts in the SYSOBDS catalog table during the materialization of pending definition changes.

When MODIFY RECOVERY is run on an entire table space, MODIFY RECOVERY removes the SYSOBDS entries after deletion of the last image copy that contains version 0 data rows or keys for the table space or associated indexes.

Related tasks:
- Materializing pending definition changes (Db2 Administration Guide)

Related reference:
- SYSIBM.SYSLGRNX table (Db2 SQL)
- SYSIBM.SYSCOPY table (Db2 SQL)
- “COPY-pending status” on page 1040

Related information:
- DSNU572I (Db2 Messages)

Reclaiming space in the DBD

You can reclaim space in the DBD when you drop a table by using the MODIFY RECOVERY utility.

Procedure

To reclaim space in the DBD when you drop a table:
1. Commit the drop.
2. Run the REORG utility.
3. Run the COPY utility to make a full image copy of the table space.
4. Run the MODIFY RECOVERY utility with the DELETE or RETAIN option to delete all previous image copies.

Improving REORG performance after adding a column

After you add a column to a table space, you can take certain steps to improve performance.

About this task

After a column is added to a table space, the next REORG utility job of that table space creates default values for the new column, as follows:

- During its UNLOAD phase, the REORG job creates default values by converting all fields in each row to the external Db2 format.
- During the RELOAD phase, the REORG job then converts the default values to the internal Db2 format.

This REORG processing, referred to as a compression cycle, occurs on each subsequent run of the REORG utility on this table space. You can improve performance by avoiding the compression cycle each time that the REORG job runs on the table space.
Procedure

To improve performance after adding a column to a table space:
1. Run the REORG utility on the table space.
2. Run the COPY utility to make a full image copy of the table space.
3. Run MODIFY RECOVERY with the DELETE or RETAIN option to delete all previous image copies. MODIFY RECOVERY changes the status of the column that is added after using the ALTER statement only if SYSCOPY rows need to be deleted.

The effect of MODIFY RECOVERY on version numbers

When you run MODIFY RECOVERY, the utility updates the range of used version numbers for table spaces and for indexes that are defined with the COPY YES attribute.

MODIFY RECOVERY updates the OLDEST_VERSION column of the appropriate catalog table or tables with the version number of the oldest version that has not yet been applied to the entire object.

If a SYSCOPY record is deleted that has an OLDEST_VERSION number that equals the CURRENT_VERSION number of the table space or index, MODIFY RECOVERY updates the OLDEST_VERSION number in the appropriate catalog table or tables with the CURRENT_VERSION number.

Db2 can reuse any version numbers that are not in the range that is set by the values in the OLDEST_VERSION and CURRENT_VERSION columns.

Db2 stores the range of used version numbers in the OLDEST_VERSION and CURRENT_VERSION columns of one or more of the following catalog tables, depending on the object:
- SYSIBM.SYSTABLESPACE
- SYSIBM.SYSTABLEPART
- SYSIBM.SYSINDEXES
- SYSIBM.SYSINDEXPART

The OLDEST_VERSION column contains the oldest used version number, and the CURRENT_VERSION column contains the current version number.

Recycling of version numbers is required when all of the version numbers are being used. All version numbers are being used when one of the following situations is true:
- The value in the CURRENT_VERSION column is one less than the value in the OLDEST_VERSION column.
- The value in the CURRENT_VERSION column is 255 for table spaces or 15 for indexes, and the value in the OLDEST_VERSION column is 0 or 1.

To recycle version numbers for indexes that are defined with the COPY NO attribute, run LOAD REPLACE, REBUILD INDEX, REORG INDEX, or REORG TABLESPACE.

Related concepts:
- Table space versions (Db2 Administration Guide)
Sample MODIFY RECOVERY control statements

Use the sample control statements as models for developing your own MODIFY RECOVERY control statements.

**Example 1: Deleting SYSCOPY and SYSLGRNX records that are over a certain age**

The following control statement specifies that the MODIFY RECOVERY utility is to delete all SYSCOPY and SYSLGRNX records that are older than 90 days for table space DSN8D81ADSN8S81E.

```
//STEP1 EXEC DSNUPROC,UID='IUIQU2UD.MODRCV1',
     // UTPROC='',SYSTEM='DSN'
//SYSIN DD *
MODIFY RECOVERY TABLESPACE DSN8D81A.DSN8S81E DELETE AGE(90)
/*
```

**Example 2: Deleting SYSCOPY and SYSLGRNX records that are older than a certain date**

The following control statement specifies that MODIFY RECOVERY is to delete all SYSCOPY and SYSLGRNX records that were written before 10 September 2002.

```
MODIFY RECOVERY TABLESPACE DSN8D11A.DSN8S11E DELETE DATE(20020910)
```  

**Example 3: Deleting SYSCOPY records for partitions**

The following control statements specifies that MODIFY RECOVERY is to delete the following SYSCOPY records for table space TU5AP053:

- Any records in partition 2 that are older than 5 days
- Any records in partition 3 that were written before 9 October 2006

```
//STEP2 EXEC DSNUPROC,UID='FUN5U053.STEP2',UTPROC='',SYSTEM='SSTR'
//SYSIN DD *

MODIFY RECOVERY TABLESPACE TU5AP053
    DSNUM 2
    DELETE AGE(5)

MODIFY RECOVERY TABLESPACE TU5AP053
    DSNUM 3
    DELETE DATE(061009)
/*
```

*Figure 41. Example MODIFY RECOVERY statements that delete SYSCOPY records for partitions*

**Example 4: Deleting all SYSCOPY records for objects in a list and viewing the results**

In the following example job, the LISTDEF utility control statements define three lists (L1, L2, L3). The first group of REPORT utility control statements then specify that the utility is to report recovery information for the objects in these lists. Next, the MODIFY RECOVERY control statement specifies that the utility is to delete all SYSCOPY records for the objects in the L1 list. Finally, the second group of REPORT control statements specify that the utility is to report the recovery information for the same three lists. In this second report, no information will be reported for the objects in the L1 list because all of the SYSCOPY records have
been deleted.

```bash
//STEP4 EXEC DSNUPROC,UID='JULTU224.RCV1',
// UTPROC='',SYSTEM='SSTR'
//SYSIN DD *
LISTDEF L1 INCLUDE TABLESPACE DBLT2401.T*
LISTDEF L2 INCLUDE INDEXSPACE DBLT2401.I*
LISTDEF L3 INCLUDE INDEX IXLT2402
REPORT RECOVERY TABLESPACE LIST L1
REPORT RECOVERY INDEXSPACE LIST L2
REPORT RECOVERY INDEX LIST L3
MODIFY RECOVERY LIST L1
   DELETE DATE(*)
REPORT RECOVERY TABLESPACE LIST L1
REPORT RECOVERY INDEXSPACE LIST L2
REPORT RECOVERY INDEX LIST L3
/*
Figure 42. Example MODIFY RECOVERY statement that deletes all SYSCOPY records
```

**Example 5: Retaining SYSCOPY and SYSLGRNX records of a GDG**

The following control statement specifies that MODIFY RECOVERY is to retain as many recent records in SYSIBM.SYSCOPY as defined in the GDG limit.

MODIFY RECOVERY TABLESPACE DBKQBL01.TPKQBL01 RETAIN GDGLIMIT

**Example 6: Retaining SYSCOPY and SYSLGRNX records**

The following control statement specifies that MODIFY RECOVERY is to retain 4 recent records in SYSIBM.SYSCOPY.

MODIFY RECOVERY TABLESPACE DBKQBL01.TPKQBL01 RETAIN LAST (4)

**Example 7: Deleting SYSCOPY and SYSLGRNX records for clone objects**

The following control statement specifies that MODIFY RECOVERY is to delete SYSCOPY records and SYSLGRNX records for only clone objects.

MODIFY RECOVERY TABLESPACE DBKQBL01.TPKQBL01
   CLONE
   DELETE AGE(*)

**Related reference:**

Chapter 18, “LISTDEF,” on page 215
Chapter 30, “REPORT,” on page 703
Chapter 22. MODIFY STATISTICS

The MODIFY STATISTICS online utility deletes unwanted statistics history records from the corresponding catalog tables. You can remove statistics history records that were written before a specific date, or you can remove records of a specific age. You can delete records for an entire table space, index space, or index.

Run MODIFY STATISTICS regularly to clear outdated information from the statistics history catalog tables. By deleting outdated information from those tables, you can improve performance for processes that access data from those tables.

Restriction: MODIFY STATISTICS does not delete statistics history records for clone tables because statistics are not collected for these tables.

Output

MODIFY STATISTICS deletes rows from the following catalog tables:

- SYSIBM.SYSCOLDIST_HIST
- SYSIBM.SYSCOLUMNS_HIST
- SYSIBM.SYSINDEXES_HIST
- SYSIBM.SYSINDEXPART_HIST
- SYSIBM.SYSINDEXSTATS_HIST
- SYSIBM.SYSLOBSTATS_HIST
- SYSIBM.SYSTABLEPART_HIST
- SYSIBM.SYSTABLESTATS_HIST
- SYSIBM.SYSTABLES_HIST
- SYSKEYTARGETS_HIST
- SYSKEYTGTDIST_HIST

Authorization required

To execute this utility, you must use a privilege set that includes one of the following authorities:

- STATS privilege for the database to run MODIFY STATISTICS.
- DBADM, DBCTRL, or DBMAINT authority for the database. If the object on which the utility operates is in an implicitly created database, DBADM authority on the implicitly created database or DSNDB04 is required.
- SQLADM authority.
- System DBADM authority.
- SYSCTRL or SYSADM authority.

A user ID with installation SYSOPR authority can also execute MODIFY STATISTICS, but only on a table space in the DSNDB01 or DSNDB06 database.

Execution phases of MODIFY STATISTICS

The MODIFY STATISTICS utility operates in these phases:

<table>
<thead>
<tr>
<th>Phase</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTILINIT</td>
<td>Performs initialization and setup</td>
</tr>
</tbody>
</table>
MODIFYS
Deletes records

UTILTERM
Performs cleanup

Syntax and options of the MODIFY STATISTICS control statement

The MODIFY STATISTICS utility control statement, with its multiple options, defines the function that the utility job performs.

You can create a control statement with the ISPF/PDF edit function. After creating it, save it in a sequential or partitioned data set. When you create the JCL for running the job, use the SYSIN DD statement to specify the name of the data set that contains the utility control statement.

Syntax diagram

Option descriptions

LIST listdef-name
Specifies the name of a previously defined LISTDEF list name. You cannot repeat the LIST keyword or specify it with TABLESPACE, INDEXSPACE, or INDEX.

The list can contain index spaces, table spaces, or both. MODIFY STATISTICS is invoked once for each object in the list.

The partitions or partition ranges can be specified in a list.

TABLESPACE database-name.table-space-name
Specifies the database and the table space for which catalog history records are to be deleted.

database-name
Specifies the name of the database to which the table space belongs.

database-name is optional.

The default value is DSNDB04.

table-space-name
Specifies the name of the table space for which statistics are to be deleted.
INDEXSPACE database-name.index-space-name
Specifies the qualified name of the index space for which catalog history information is to be deleted. The utility lists the name in the SYSIBM.SYSINDEXES table.

database-name
Optionally specifies the name of the database to which the index space belongs.

The default value is DSNDDB04.

index-space-name
Specifies the name of the index space for which the statistics are to be deleted.

INDEX creator-id.index-name
Specifies the index for which catalog history information is to be deleted.

creator-id
Optionally specifies the creator of the index.

The default value is DSNDDB04.

index-name
Specifies the name of the index for which the statistics are to be deleted. Enclose the index name in quotation marks if the name contains a blank.

DELETE
Indicates that records are to be deleted.

ALL
Deletes all statistics history rows that are related to the specified object from all catalog history tables.

Rows from the following history tables are deleted only when you specify DELETE ALL:
• SYSTABLES_HIST
• SYTABSTATS_HIST
• SYSINDEXES_HIST
• SYSINDEXSTATS_HIST
• SYSKEYTARGETS_HIST

ACCESSPATH
Deletes all access-path statistics history rows that are related to the specified object from the following history tables:
• SYSIBM.SYSCOLDIST_HIST
• SYSIBM.SYSCOLUMNS_HIST
• SYSKEYTGTDIST_HIST

SPACE
Deletes all space-tuning statistics history rows that are related to the specified object from the following history tables:
• SYSIBM.SYSINDEXPART_HIST
• SYSIBM.SYSTABLEPART_HIST
• SYSIBM.SYSLOBSTATS_HIST

AGE (integer)
Deletes all statistics history rows that are related to the specified object and that are older than a specified number of days.
(integer)
  Specifies the number of days in a range 0 - 32 767. This option cannot
delete records that are created today (age 0).

(*)
  Deletes all records, regardless of their age.

DATE (integer)
  Deletes all statistics history rows that were written before a specified date.

(integer)
  Specifies the date in an eight-character format. Specify a year (yyyy), month
(mm), and day (dd) in the form yyyyymmdd.

(*)
  Deletes all records, regardless of the date on which they were written.

Data sets that MODIFY STATISTICS uses

The MODIFY STATISTICS utility uses a number of data sets during its operation.

The following table lists the data sets that MODIFY STATISTICS uses. The table
lists the DD name that is used to identify the data set, a description of the data set,
and an indication of whether it is required. Include statements in your JCL for each
required data set and any optional data sets that you want to use.

Table 55. Data sets that MODIFY STATISTICS uses

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSIN</td>
<td>Input data set that contains the utility control statement</td>
<td>Yes</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>Output data set for messages</td>
<td>Yes</td>
</tr>
</tbody>
</table>

The following object is named in the utility control statement and does not require
a DD statement in the JCL:

Table space or index space
  Object for which records are to be deleted.

Related concepts:
  “Data sets that online utilities use” on page 16

Concurrency and compatibility for MODIFY STATISTICS

The MODIFY STATISTICS utility has certain concurrency and compatibility
characteristics associated with it.

Db2 treats individual data and index partitions as distinct target objects. Utilities
that operate on different partitions of the same table space or index space are
compatible.

The following table shows the restrictive state that the utility sets on the target
object.

Table 56. Claim classes of MODIFY STATISTICS operations.

<table>
<thead>
<tr>
<th>Target</th>
<th>MODIFY STATISTICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table space, index, or index space</td>
<td>UTRW</td>
</tr>
</tbody>
</table>
Table 56. Claim classes of MODIFY STATISTICS operations. (continued)

<table>
<thead>
<tr>
<th>Target</th>
<th>MODIFY STATISTICS</th>
</tr>
</thead>
</table>

Legend:

UTRW - Utility restrictive state - read-write access allowed.

Guidelines for deciding which statistics history rows to delete

After analyzing trends by using the relevant historical catalog information and possibly taking actions based on this information, consider deleting all or part of the statistics history catalog rows.

Deleting outdated information from the statistics history catalog tables can improve performance for processes that access data from those tables. You also make available the space in the catalog. Then, the next time you update the relevant statistics by using RUNSTATS TABLESPACE, REBUILD INDEX, or REORG INDEX, Db2 repopulates the statistics history catalog tables with more recent historical data. Examining this data lets you determine the efficacy of any adjustments that you made as a result of your previous analysis.

Be aware that when you manually insert, update, or delete catalog information, Db2 does not store the historical information for those operations in the historical catalog tables.

Deletion of specific statistics history rows

The MODIFY STATISTICS utility lets you delete some or all statistics history rows for a table space, an index space, or an index.

You can choose to delete only the statistics rows that relate to access path selection by specifying the ACCESSPATH option. Alternatively, you can delete the rows that relate to space statistics by using the SPACE option. To delete rows in all statistics history catalog tables, including the SYSIBM.SYSTABLES_HIST catalog table, you must specify the DELETE ALL option in the utility control statement.

To delete statistics from the RUNSTATS history tables, you can either use the MODIFY STATISTICS utility or issue SQL DELETE statements. The MODIFY STATISTICS utility simplifies the purging of old statistics without requiring you to write the SQL DELETE statements. You can also delete rows that meet the age and date criteria by specifying the corresponding keywords (AGE and DATE) for a particular object.

To avoid time outs when you delete historical statistics with MODIFY STATISTICS, you should increase the LOCKMAX parameter for DSNDB06.SYSHIST with ALTER TABLESPACE.

Termination or restart of MODIFY STATISTICS

You can terminate and restart the MODIFY STATISTICS utility.

You can use the TERM UTILITY command to terminate the MODIFY STATISTICS utility in any phase.
You can restart a MODIFY STATISTICS utility job, but it starts from the beginning again.

Related tasks:
“Restarting an online utility” on page 44

Sample MODIFY STATISTICS control statements

Use the sample control statements as models for developing your own MODIFY STATISTICS control statements.

Example 1: Deleting SYSIBM.SYSTABLES_HIST records by age

The following control statement specifies that the MODIFY STATISTICS utility is delete all statistics history records for table space DSN8D81A.DSN8S81E that are older than 60 days.

```
//STEP1 EXEC DSNUPROC,UID='IUIQU2UD.MODSTAT1',
    // UTPROC='',SYSTEM='DSN'
//SYSIN DD *
    MODIFY STATISTICS TABLESPACE DSN8D81A.DSN8S81E
    DELETE ALL
    AGE 60
/*
```

Example 2: Deleting access path records for all objects in a list

The following MODIFY STATISTICS control statement specifies that the utility is to delete access-path statistics history rows that were created before 17 April 2000 for objects in the specified list. The list, M1, is defined in the preceding LISTDEF control statement and includes table spaces DB0E1501.TL0E1501 and DSN8D81A.DSN8S81E.

```
//STEP9 EXEC DSNUPROC,UID='JUEU115.MDFYL9',
    // UTPROC='',
    // SYSTEM='SSTR'
//SYSPRINT DD SYSOUT=* 
//SYSIN DD *
    LISTDEF M1 INCLUDE TABLESPACE DB0E1501.TL0E1501
    INCLUDE TABLESPACE DSN8D81A.DSN8S81E
    MODIFY STATISTICS LIST M1
    DELETE ACCESSPATH DATE(20000417)
/*
```

Figure 43. MODIFY STATISTICS control statement that specifies that access path history records are to be deleted

Example 3: Deleting space-tuning statistics records for an index by age

The following control statement specifies that MODIFY STATISTICS is to delete space-tuning statistics records for index ADMF001.IXOE15S1 that are older than one day.
Example 4: Deleting all statistics history records for an index space

The following control statement specifies that MODIFY STATISTICS is to delete all statistics history records for index space DBOE1501.IUOE1501. Note that the deleted records are not limited by date because (*) is specified.

//STEP9 EXEC DSNUPROC,UID='JUOEU115.MOFYS9',
//       UTPROC='',
//       SYSTEM='SSTR'
//SYSPRINT DD SYSOUT=*  
//SYSIN   DD *

   MODIFY STATISTICS INDEX ADMF001.IXOE15S1
       DELETE SPACE AGE 1
/*
Figure 44. MODIFY STATISTICS control statement that specifies that space-tuning statistics records are to be deleted

Example 4: Deleting all statistics history records for an index space

The following control statement specifies that MODIFY STATISTICS is to delete all statistics history records for index space DBOE1501.IUOE1501. Note that the deleted records are not limited by date because (*) is specified.

//STEP8 EXEC DSNUPROC,UID='JUOEU115.MDFYL8',
//       UTPROC='',
//       SYSTEM='SSTR'
//SYSPRINT DD SYSOUT=*  
//SYSIN   DD *

   MODIFY STATISTICS INDEXSPACE DBOE1501.IUOE1501
       DELETE ALL DATE (*)
/*
Figure 45. MODIFY STATISTICS control statement that specifies that all statistics history records are to be deleted
The OPTIONS online utility control statement specifies processing options that are applicable across many utility executions in a job step.

By specifying various options, you can:
- Preview utility control statements
- Preview LISTDEF or TEMPLATE definitions
- Override library names for LISTDEF lists or TEMPLATE definitions
- Specify how to handle errors during list processing
- Alter the return code for warning messages
- Restore all default options

You can repeat an OPTIONS control statement within the SYSIN DD statement. If you repeat the control statement, it entirely replaces any prior OPTIONS control statement.

Output

The OPTIONS control statement sets the specified processing options for the duration of the job step, or until replaced by another OPTIONS control statement within the same job step.

Authorization required

The OPTIONS control statement performs setup for subsequent control statements. The OPTIONS statement itself requires no privileges to execute.

Execution phases of OPTIONS

The OPTIONS control statement executes entirely in the UTILINIT phase, in which it performs setup for the subsequent utility.

Syntax and options of the OPTIONS control statement

The OPTIONS utility control statement, with its multiple options, defines the function that the utility job performs.

You can create a control statement with the ISPF/PDF edit function. After creating it, save it in a sequential or partitioned data set. When you create the JCL for running the job, use the SYSIN DD statement to specify the name of the data set that contains the utility control statement.

Syntax diagram
Option descriptions

PREVIEW

Specifies that the utility control statements that follow are to run in PREVIEW mode. The utility checks for syntax errors in all utility control statements, but normal utility execution does not take place. If the syntax is valid, the utility expands all LISTDEF lists and TEMPLATE DSNs that appear in SYSIN and prints results to the SYSPRINT data set.

PREVIEW evaluates and expands all LISTDEF statements into an actual list of table spaces or index spaces. It evaluates TEMPLATE DSNs and uses variable substitution for actual data set names when possible. It also expands lists from the SYSLISTD DD and TEMPLATE DSNs from the SYSTEMPL DD that a utility invocation references.

A definitive preview of TEMPLATE DSN values is not always possible. Substitution values for some variables, such as &DATE., &TIME., &SEQ. and &PART., can change at execution time. In some cases, PREVIEW generates approximate data set names. The OPTIONS utility substitutes unknown character variables with the character string "UNKNOWN" and unknown integer variables with zeroes.

Instead of OPTIONS PREVIEW, you can use a JCL PARM to activate preview processing. Although the two functions are identical, use JCL PARM to preview an existing set of utility control statements. Use the OPTION PREVIEW control statement when you invoke Db2 utilities through a stored procedure.

The JCL PARM is specified as the third JCL PARM of DSNUTILB and on the UTPROC variable of DSNUPROC, as shown in the following JCL:

```
//STEP1 EXEC DSNUPROC, UID='JULY106.RECOVE1',
//   UTPROC='PREVIEW', SYSTEM='SSTR'
```

The PARM value PREVIEW causes the utility control statements in that job step to be processed for preview only. The LISTDEF and TEMPLATE control statements are expanded, but the utility does not execute.
OPTIONS PREVIEW is identical to the PREVIEW JCL parameter, except that you can specify a subsequent OPTIONS statement to turn off the preview for OPTIONS PREVIEW. Absence of the PREVIEW keyword in the OPTION control statement turns off preview processing, but it does not override the PREVIEW JCL parameter, which, if specified, remains in effect for the entire job step.

LISTDEFDD  ddname
  Specifies the ddname of the LISTDEF definition library. A LISTDEF library is a data set that contains only LISTDEF utility control statements. This data set is processed only when a referenced LIST is not found in SYSIN.

  The default value is SYSLISTD.

TEMPLATEDD  ddname
  Specifies the ddname of the TEMPLATE definition library. A TEMPLATE library is a data set that contains only TEMPLATE utility control statements. This data set is processed only when a referenced name does not exist in the job step as a DD name and is not found in SYSIN as a TEMPLATE name.

  The default value is SYSTEMPL.

FILSZ  integer
  Specifies a file size in megabytes and overrides the file size for the sort program when sort work data sets are allocated by the utility with system parameter UTSORTAL set to YES. Only use this keyword under the direction of IBM Support.

EVENT
  Specifies one or more pairs of utility processing events and the matching action for the event. Not all actions are valid for all events.

  The parentheses and commas in the EVENT operand are currently optional but they may be required in a future release.

ITEMERROR
  Specifies how utility processing is to handle errors during list processing. Specifically, this keyword indicates the effect on processing in response to return code 8. By default, utility processing stops (HALT). The ITEMERROR event does not include abnormal terminations (abends).

  Note that for the QUIESCE utility, the indexes for the table spaces in the list, if any, are considered as list items for the purposes of the ITEMERROR event. ITEMERROR affects how errors are handled on both the table spaces and the indexes.

HALT
  Specifies that the utility is to stop after the event.

SKIP
  Ignores the event and skips the list item. Processing continues with the next item in the list.

  SKIP applies only during the processing of a valid list. SKIP does not apply if a utility detects that a list is not valid for the utility that is invoked. In that case, the list is rejected with an error message and the processing of the list is not initiated.

  If any of the items in a list is skipped, the utility produces a return code of 8, which terminates the job step. The following code shows an OPTIONS statement with the SKIP option:
OPTIONS EVENT (ITEMERROR, SKIP)
COPY LISTA
COPY LISTB

If LISTA contains ten objects and one object produces a return code 8 during the COPY, the other nine objects in the list are copied successfully. The job step ends with a return code 8 and COPY LISTB is not executed.

WARNING
Specifies a response to the return code message event.

Use WARNING to alter the return code for warning messages. You can alter the return code from message DSNU010I with this option. If you alter the message return code, message DSNU1024I is issued to document the new return code.

Action choices are as follows:

RC0
Lowers the final return code of a single utility invocation that ends in a return code 4 to a return code of 0. Use RC0 to force a return code of 0 for warning messages.

Use this option only when return code 4 is expected, is acceptable, and other mechanisms are in place to validate the results of a utility execution.

RC4
Specifies that return codes for warning messages are to remain unchanged. Use RC4 to override a previous OPTIONS WARNING specification in the same job step.

RC8
Raises the final return code of a single utility invocation that ends in a return code 4 to a return code of 8. Use RC8 to force a return code of 8 for warning messages. The return code of 8 causes the job step to terminate and subsequent utility control statements are not executed.

OFF
Specifies that all default options are to be restored. OPTIONS OFF does not override the PREVIEW JCL parameter, which, if specified, remains in effect for the entire job step. You cannot specify any other OPTIONS keywords with OPTIONS OFF.

OPTIONS OFF is equivalent to OPTIONS LISTDEFDD SYSLISTD TEMPLATEDDD SYSTEMPL EVENT (ITEMERROR, HALT, WARNING, RC4).

KEY
Specifies an option that you should use only when you are instructed by IBM Support. OPTIONS KEY is followed by a single operand that IBM Support provides when needed.

Related reference:

Db2 Sort for z/OS

Related information:

DFSORT Application Programming Guide

Concurrency and compatibility for OPTIONS

The OPTIONS utility has certain concurrency and compatibility characteristics associated with it.
OPTIONS is a utility control statement that you can use to set up an environment for another utility to follow. The OPTIONS statement is stored until a specific utility references the statement. When referenced by another utility, the list is expanded. At that time, the concurrency and compatibility restrictions of that utility apply, with the additional restriction that the catalog tables that are necessary to expand the list must be available for read-only access.

**Executing statements in preview mode**

You can execute utility control statements in preview mode. The utility checks for syntax errors in all utility control statements, but normal utility execution does not occur.

**About this task**

Statistics profiles can be previewed using the PREVIEW option. Under normal execution, statistics profiles are stored in the SYSIBM.SYSTABLES_PROFILES catalog table. When executing RUNSTATS with the PREVIEW option, Db2 only prints the statistics profile for each table to SYSPRINT and normal utility execution does not take place.

Please note that the profile text is displayed prior to parsing for syntactical errors.

The contents of the profile is displayed using DSNU1376I.

**Procedure**

To execute utility control statements in preview mode:

Specify the PREVIEW option in the OPTIONS control statement. Control statements are previewed for use with LISTDEF lists and TEMPLATE definitions but the specified options are not actually executed.

**Related reference:**

“RUNSTATS TABLESPACE syntax and options” on page 743

**Specifying LISTDEF and TEMPLATE libraries**

You can override the names of the optional library data sets.

**Procedure**

To specify LISTDEF and TEMPLATE libraries:

Specify the LISTDEFDD option and the TEMPLATDD option in the OPTIONS control statement to override the names of the optional library data sets.

**Related tasks:**

“Creating LISTDEF libraries” on page 230

**Related reference:**

Chapter 18, “LISTDEF,” on page 215

**Overriding standard utility processing behavior**

You can alter settings for warning return codes and error handling during list processing.
Procedure

To override standard utility processing behavior:

Specify the EVENT option in the OPTIONS control statement.

Termination or restart of OPTIONS

You can terminate and restart the OPTIONS utility.

You can terminate an OPTIONS utility job by using the TERM UTILITY command if you submitted the job or have SYSOPR, SYSCTRL, or SYSADM authority.

You can restart an OPTIONS utility job, but it starts from the beginning again. If you are restarting this utility as part of a larger job in which OPTIONS completed successfully, but a later utility failed, do not change the OPTIONS utility control statement, if possible. If you must change the OPTIONS utility control statement, use caution; any changes can cause the restart processing to fail. For example, if you specify a valid OPTIONS statement in the initial invocation, and then on restart, specify OPTIONS PREVIEW, the job fails.

Related tasks:

“Restarting an online utility” on page 44

Sample OPTIONS control statements

Use the sample control statements as models for developing your own OPTIONS control statements.

Example 1: Checking control statement syntax and previewing lists and TEMPLATE data set names

The following OPTIONS utility control statement specifies that the subsequent utility control statements are to run in PREVIEW mode. In PREVIEW mode, Db2 checks for syntax errors in all utility control statements, but normal utility execution does not take place. If the syntax is valid, Db2 expands the CPYLIST list and the data set names in the COPYLOC and COPYREM TEMPLATE utility control statements and prints these results to the SYSPRINT data set.

```
OPTIONS PREVIEW
TEMPLATE COPYLOC UNIT(SYSDA)
   DSN(&DB..&TS..&JDATE..&STEPNAME..COPY&IC..&LOCREM..&PB.)
   DISP(NEW,CATLG,CATLG) SPACE(200,20) TRK
   VOLUMES(SCR03)
TEMPLATE COPYREM UNIT(SYSDA)
   DSN(&DB..&TS..&UT..&TIME..COPY&IC..&LOCREM..&PB.)
   DISP(NEW,CATLG,CATLG) SPACE(100,10) TRK
LISTDEF CPYLIST INCLUDE TABLESPACES DATABASE DBLT0701
COPY LIST CPYLIST FULL YES
   COPYDDN(COPYLOC,COPYLOC)
   RECOVERYDDN(COPYREM,COPYREM)
   SHRLEVEL REFERENCE
```

Figure 46. Example OPTIONS statement for checking syntax and previewing lists and templates.
Example 2: Specifying LISTDEF and TEMPLATE definition libraries

In the following example, the OPTIONS control statements specify the DD names of the LISTDEF definition libraries and the TEMPLATE definition libraries.

The first OPTIONS statement specifies that the LISTDEF definition library is identified by the V1LIST DD statement and the TEMPLATE definition library is identified by the V1TEMPL DD statement. These definition libraries apply to the subsequent COPY utility control statement. Therefore, if Db2 does not find the PAYTBSP list in SYSIN, it searches the V1LIST library, and if Db2 does not find the PAYTEMP1 template in SYSIN, it searches the V1TEMP library.

The second OPTIONS statement is similar to the first, but it identifies different libraries and applies to the second COPY control statement. This second COPY control statement looks similar to the first COPY job. However, this statement processes a different list and uses a different template. Whereas the first COPY job uses the PAYTBSP list from the V1LIST library, the second COPY job uses the PAYTBSP list from the V2LIST library. Also, the first COPY job uses the PAYTEMP1 template from the V1TEMPL library, the second COPY job uses the PAYTEMP1 template from the V2TEMPL library.

OPTIONS LISTDEFDD V1LIST TEMPLATEDD V1TEMPL
COPY LIST PAYTBSP COPYDDN(PAYTEMP1,PAYTEMP1)

OPTIONS LISTDEFDD V2LIST TEMPLATEDD V2TEMPL
COPY LIST PAYTBSP COPYDDN(PAYTEMP1,PAYTEMP1)

Example 3: Forcing a return code 0

In the following example, the first OPTIONS control statement forces a return code of 0 for the subsequent MODIFY RECOVERY utility control statement. Ordinarily, this statement ends with a return code of 4 because it specifies that Db2 is to delete all SYSCOPY and SYSLGRNX records for table space A.B. The second OPTIONS control statement restores the default options, so that no return codes will be overridden for the second MODIFY RECOVERY control statement.

OPTIONS EVENT(WARNING,RC0)
MODIFY RECOVERY TABLESPACE A.B DELETE AGE(*)
OPTIONS OFF
MODIFY RECOVERY TABLESPACE C.D DELETE AGE(30)

Example 4: Checking syntax and skipping errors while processing list objects

In the following control statement, the first OPTIONS utility control statement specifies that the subsequent utility control statements are to run in PREVIEW mode. In PREVIEW mode, Db2 checks for syntax errors in all utility control statements, but normal utility execution does not take place. If the syntax is valid, Db2 expands the three lists (LIST1_LISTDEF, LIST2_LISTDEF, and LIST3_LISTDEF) and prints these results to the SYSPRINT data set.

The second OPTIONS control statement specifies how Db2 is to handle return codes of 8 in any subsequent utility statements that process a valid list. If processing of a list item produces return code 8, Db2 skips that item, and continues to process the rest of the items in the list, but Db2 does not process the next utility control statement. Instead, the job ends with return code 8.
OPTIONS PREVIEW

LISTDEF COPY1_LISTDEF
  INCLUDE TABLESPACES TABLESPACE DSNDB01.SPT01
  INCLUDE TABLESPACES TABLESPACE DSNDB06.SYSGPAUT
  INCLUDE TABLESPACES TABLESPACE DBA91302.TFA9132*

LISTDEF COPY2_LISTDEF
  INCLUDE TABLESPACES TABLESPACE DBA91303.TLA9133A
  INCLUDE TABLESPACES TABLESPACE DBA91303.TSA9133B
  INCLUDE TABLESPACES TABLESPACE DBA91303.TPA9133C
  INCLUDE TABLESPACES TABLESPACE DBA91304.TLA9134A
  INCLUDE TABLESPACES TABLESPACE DSNDB06.SYSUSER
  INCLUDE TABLESPACES TABLESPACE DSNDB06.SYSGRTNS
  INCLUDE TABLESPACES TABLESPACE DSNDB06.SYSDDF

LISTDEF COPY3_LISTDEF
  INCLUDE TABLESPACES TABLESPACE DBA91304.TSA9134B
  INCLUDE TABLESPACES TABLESPACE DSNDB06.SYSHIST
  INCLUDE TABLESPACES TABLESPACE DSNDB06.SYSGRTNS
  INCLUDE TABLESPACES TABLESPACE DSNDB06.SYSJAVA
  INCLUDE TABLESPACES TABLESPACE DBA91304.TPA9134C

OPTIONS EVENT(ITEMERROR,SKIP)
TEMPLATE TMP1 UNIT(SYSDA) DISP(MOD,CATLG,CATLG)
  VOLUMES(SCR03)
  DSN(DH109013...COPY&ICTYPE.&LOCREM.&PRIBAC.)

COPY LIST COPY1_LISTDEF SHRLEVEL REFERENCE
COPYDDN (TMP1)
  RECOVERYDDN (TMP1)
  FULL YES

COPY LIST COPY2_LISTDEF SHRLEVEL REFERENCE
COPYDDN (TMP1,TMP1)
  FULL YES

COPY LIST COPY3_LISTDEF SHRLEVEL REFERENCE
COPYDDN (TMP1,TMP1)
  RECOVERYDDN (TMP1,TMP1)
  FULL YES
Chapter 24. QUIESCE

The QUIESCE utility establishes a quiesce point for a table space, partition, table space set, or list of table spaces and table space sets. A quiesce point is a point at which data is consistent across these objects. You can later recover a table space to its quiesce point by using the RECOVER utility.

Output

When you request that the QUIESCE utility take a quiesce point, the quiesce point is the current log RBA or log record sequence number (LRSN). QUIESCE then records the quiesce point in the SYSIBM.SYSCOPY catalog table.

A quiesce point is not essential when you plan for point-in-time recoveries. The RECOVER utility can recover data to a prior point-in-time with consistency without a quiesce point. The utility can recover objects with transactional consistency, which means that the objects contain only data that has been committed. However, recovering objects to a quiesce point can be faster because no work must be backed out. You might also want to establish quiesce points for related sets of objects if you need to plan for a point-in-time recovery for the entire set.

Related information:
- "Point-in-time recovery” on page 495
- "Common quiesce points” on page 421

With the WRITE(YES) option, QUIESCE writes changed pages for the table spaces and their indexes from the Db2 buffer pool to disk. The catalog table SYSCOPY records the current RBA and the timestamp of the quiesce point. A row with ICTYPE='Q' is inserted into SYSIBM.SYSCOPY for each table space that is quiesced. Db2 also inserts a SYSCOPY row with ICTYPE='Q' for any indexes (defined with the COPY YES attribute) over a table space that is being quiesced. (Table spaces DSNDB06.SYSTSCPY, DSNDB01.DBD01, DSNDB01.SYSUTILX, and DSNDB01.SYSDBDXA are an exception; their information is written to the log.)

Authorization required

To execute this utility, you must use a privilege set that includes one of the following authorities:

- IMAGCOPY privilege for the database
- DBADM, DBCTRL, or DBMAINT authority for the database. If the object on which the utility operates is in an implicitly created database, DBADM authority on the implicitly created database or DSNDB04 is required.
- System DBADM authority
- SYSCTRL or SYSADM authority

An ID with installation SYSOPR authority can also execute QUIESCE, but only on a table space in the DSNDB01 or DSNDB06 database.

You can specify DSNDB01.SYSUTILX, but you cannot include it in a list with other table spaces to be quiesced. Recovery to the current catalog and directory table spaces is preferred and recommended. However, if you want a point-in-time
recovery of the catalog and directory table spaces, a separate quiesce of DSNDB06.SYSTSCPY is required after a quiesce of the other catalog and directory table spaces.

Execution phases of QUIESCE

The QUIESCE utility operates in these phases:

<table>
<thead>
<tr>
<th>Phase</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTILINIT</td>
<td>Initialization and setup</td>
</tr>
<tr>
<td>QUIESCE</td>
<td>Determining the quiesce point and updating the catalog</td>
</tr>
<tr>
<td>UTILTERM</td>
<td>Cleanup</td>
</tr>
</tbody>
</table>

Syntax and options of the QUIESCE control statement

The QUIESCE utility control statement, with its multiple options, defines the function that the utility job performs.

You can create a control statement with the ISPF/PDF edit function. After you create the statement, save it in a sequential or partitioned data set. When you create the JCL for running the job, use the SYSIN DD statement to specify the name of the data set that contains the utility control statement.

Syntax diagram

```
QUIESCE
  LIST listdef-name
  TABLESPACE database-name table-space-name
  PART integer
  TABLESPACESET database-name table-space-name
  WRITE YES
  WRITE NO
  CLONE
```

Option descriptions

The purpose of most of the QUIESCE control statement options is to specify which objects to quiesce. You can specify as many objects in your QUIESCE job as allowed by available memory in the batch address space and in the Db2 DBM1 address space. If you specify a table space more than once, utility processing continues, and the table space is quiesced only once. QUIESCE issues return code 4 and warning message DSNU533I to alert you of the duplication.

Use the following options to specify which objects to quiesce:
LIST listdef-name
Specifies the name of a previously defined LISTDEF list name that contains only table spaces. The utility allows one LIST keyword for each QUIESCE control statement. Do not specify LIST with the TABLESPACE or TABLESPACESET keyword. QUIESCE is invoked once for the entire list. For the QUIESCE utility, the related index spaces are considered to be list items for the purposes of OPTIONS ITEMERROR processing. You can alter the utility behavior during processing of related indexes with the OPTIONS ITEMERROR statement. This utility processes clone data only if the CLONE keyword is specified. The use of CLONED YES on the LISTDEF statement is not sufficient.

The partitions or partition ranges can be specified in a list.

TABLESPACE database-name.table-space-name
For QUIESCE TABLESPACE, specifies the table space that is to be quiesced.
For QUIESCE TABLESPACESET, specifies a table space in the table space set that is to be quiesced. For QUIESCE TABLESPACESET, the TABLESPACE keyword is optional.

database-name
Optionally specifies the name of the database to which the table space belongs.
The default value is DSNDB04.

table-space-name
Specifies the name of the table space that is to be quiesced. You can specify DSNDB01.SYSUTILX, but do not include that name in a list with other table spaces that are to be quiesced. If a point-in-time recovery is planned for the catalog and directory, DSNDB06.SYSTSCPY must be quiesced separately after all other catalog and directory table spaces.

All table spaces that are involved in a versioning relationship are quiesced when QUIESCE is run on either the system-period temporal table or the history table space. Auxiliary LOB and XML table spaces on both system-period temporal table spaces and history table spaces are included.

PART integer
Identifies a partition that is to be quiesced.

integer is the physical partition number. It must be in the range from 1 to the number of partitions that are defined for the table space. The maximum is 4096.

If you specify the same table space twice in a list and use PART n in one specification and PART m in the other specification, each partition is quiesced once.

TABLESPACESET
Indicates that all of the referentially related table spaces in the table space set are to be quiesced. For the purposes of the QUIESCE utility, a table space set includes the following sets of objects:
• A group of table spaces that are related through referential constraints
• A base table space with all of its LOB table spaces
• A base table space with all of its XML table spaces
• A table space with a system-period temporal table and the table space with the related history table
• A table space that includes an archive-enabled table and the table space that contains the associated archive table
Each table space set that you specify is expanded into a list of these related table spaces.

**Related information:**
- “Common quiesce points” on page 421
- Archive-enabled tables and archive tables (Introduction to Db2 for z/OS)

**CLONE**
Indicates that QUIESCE is to create a quiesce point for only the specified clone table space. This utility processes clone data only if the CLONE keyword is specified. The use of CLONED YES on the LISTDEF statement is not sufficient.

Use the following option to control the behavior of QUIESCE:

**WRITE**
- Specifies whether the changed pages from the table spaces and index spaces are to be written to disk.

**YES**
- Establishes a quiesce point and writes the changed pages from the table spaces and index spaces to disk.

**NO**
- Establishes a quiesce point but does not write the changed pages from the table spaces and index spaces to disk.

Table spaces with the NOT LOGGED attribute are not quiesced.

---

**Before running QUIESCE**

Certain activities might be required before you run the QUIESCE utility, depending on your situation.

You cannot run QUIESCE on a table space that is in COPY-pending, CHECK-pending, RECOVER-pending, or auxiliary CHECK-pending status.

**Related concepts:**
- “Resetting COPY-pending status” on page 356
- “Resetting REBUILD-pending status” on page 356

**Related tasks:**
- “Resetting CHECK-pending status” on page 97

**Related reference:**
- Appendix B, “Advisory or restrictive states,” on page 1037

---

**Data sets that QUIESCE uses**

The QUIESCE utility uses a number of data sets during its operation.

The following table lists the data sets that QUIESCE uses. The table lists the DD name that is used to identify the data set, a description of the data set, and an indication of whether it is required. Include statements in your JCL for each required data set and any optional data sets that you want to use.

**Table 57. Data sets that QUIESCE uses**

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSIN</td>
<td>Input data set that contains the utility control statement.</td>
<td>Yes</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>Output data set for messages.</td>
<td>Yes</td>
</tr>
</tbody>
</table>
The following object is named in the utility control statement and does not require a DD statement in the JCL:

**Table space**
Object that is to be quiesced. (If you want to quiesce only one partition of a table space, you must use the PART option in the control statement.)

**Related concepts:**
“Data sets that online utilities use” on page 16

**Concurrency and compatibility for QUIESCE**
The QUIESCE utility has certain concurrency and compatibility characteristics associated with it.

Db2 treats individual data and index partitions as distinct target objects. Utilities that operate on different partitions of the same table space or index space are compatible.

**Claims**
The following table shows which claim classes QUIESCE drains and any restrictive state that the utility sets on the target object.

*Table 58. Claim classes of QUIESCE operations.*

<table>
<thead>
<tr>
<th>Target</th>
<th>WRITE YES</th>
<th>WRITE NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table space or partition</td>
<td>DW/UTRO</td>
<td>DW/UTRO</td>
</tr>
<tr>
<td>Partitioning index, data-partitioned secondary index, or partition</td>
<td>DW/UTRO</td>
<td></td>
</tr>
<tr>
<td>Nonpartitioned secondary index</td>
<td>DW/UTRO</td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- DW - Drain the write claim class - concurrent access for SQL readers
- UTRO - Utility restrictive state - read-only access allowed

**Compatibility**
The following table shows which utilities can run concurrently with QUIESCE on the same target object. The target object can be a table space, an index space, or a partition of a table space or index space. If compatibility depends on particular options of a utility, that information is also documented in the table. QUIESCE does not set a utility restrictive state if the target object is DSNB01.SYSUTILX.

*Table 59. Compatibility of QUIESCE with other utilities.*

<table>
<thead>
<tr>
<th>Action</th>
<th>Compatible with QUIESCE?</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHECK DATA DELETE NO</td>
<td>Yes</td>
</tr>
<tr>
<td>CHECK DATA DELETE YES</td>
<td>No</td>
</tr>
<tr>
<td>CHECK INDEX</td>
<td>Yes</td>
</tr>
<tr>
<td>CHECK LOB</td>
<td>Yes</td>
</tr>
<tr>
<td>COPY INDEXSPACE SHRLEVEL CHANGE</td>
<td>No</td>
</tr>
<tr>
<td>COPY INDEXSPACE SHRLEVEL REFERENCE</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Table 59. Compatibility of QUIESCE with other utilities (continued)

<table>
<thead>
<tr>
<th>Action</th>
<th>Compatible with QUIESCE?</th>
</tr>
</thead>
<tbody>
<tr>
<td>COPY TABLESPACE SHRLEVEL CHANGE</td>
<td>No</td>
</tr>
<tr>
<td>COPY TABLESPACE SHRLEVEL REFERENCE</td>
<td>Yes</td>
</tr>
<tr>
<td>DIAGNOSE</td>
<td>Yes</td>
</tr>
<tr>
<td>LOAD</td>
<td>No</td>
</tr>
<tr>
<td>MERGECOPY</td>
<td>Yes</td>
</tr>
<tr>
<td>MODIFY</td>
<td>Yes</td>
</tr>
<tr>
<td>QUIESCE</td>
<td>Yes</td>
</tr>
<tr>
<td>REBUILD INDEX</td>
<td>No</td>
</tr>
<tr>
<td>RECOVER INDEX</td>
<td>No</td>
</tr>
<tr>
<td>RECOVER TABLESPACE</td>
<td>No</td>
</tr>
<tr>
<td>REORG INDEX</td>
<td>No</td>
</tr>
<tr>
<td>REORG TABLESPACE UNLOAD CONTINUE or PAUSE</td>
<td>No</td>
</tr>
<tr>
<td>REORG TABLESPACE UNLOAD ONLY or EXTERNAL</td>
<td>Yes</td>
</tr>
<tr>
<td>REPAIR DELETE or REPLACE</td>
<td>No</td>
</tr>
<tr>
<td>REPAIR DUMP or VERIFY</td>
<td>Yes</td>
</tr>
<tr>
<td>REPORT</td>
<td>Yes</td>
</tr>
<tr>
<td>RUNSTATS</td>
<td>Yes</td>
</tr>
<tr>
<td>STOSPACE</td>
<td>Yes</td>
</tr>
<tr>
<td>UNLOAD</td>
<td>Yes</td>
</tr>
</tbody>
</table>

To run the QUIESCE utility on DSND01.SYSUTILX, ensure that QUIESCE is the only utility in the job step.

QUIESCE on SYSUTILX is an exclusive job; such a job can interrupt another job between job steps, possibly causing the interrupted job to time out.

**Use of QUIESCE on catalog and directory objects**

Running the QUIESCE utility on the catalog and directory table spaces makes these objects unavailable for any updates during the utility execution. Therefore, this action can have a severe, adverse impact on your system's availability.

Running QUIESCE on the catalog and directory table spaces is not necessary to prepare for point-in-time recovery. The RECOVER utility can recover data to a point in time with consistency without a quiesce point.

If system availability is not a concern or you have planned for it, and you want a quiesce point on the catalog and directory table spaces, run QUIESCE on these table spaces. Adhere to the following requirements:

- You can quiesce DSND01.SYSUTILX, but DSND01.SYSUTILX must be the only table space in the QUIESCE utility control statement.
- A separate QUIESCE control statement for DSND06.SYSTSCPY is required after you quiesce the other catalog and directory table spaces. This separate quiesce is needed to ensure that a subsequent point-in-time recovery of
DSNDB06.SYSTSCPY recovers all of the SYSCOPY records for QUIESCE operations for the other catalog and directory objects.

Related concepts:
“Point-in-time recovery” on page 495

Common quiesce points

A common quiesce point is a point at which data is consistent across related table spaces as a result of running the QUIESCE utility. Although establishing such a quiesce point is not required for point-in-time recoveries, doing so can improve recovery time and ensure consistency for sets of related objects.

A quiesce point is not essential for point-in-time recoveries. Additional methods exist for ensuring that objects are recovered to a consistent state, without any uncommitted data. You can recover objects to any RBA or LRSN by using the TORBA or TOLOGPOINT options on the RECOVER utility statement. In this case, RECOVER automatically handles any uncommitted units of work to ensure that the data is left in a consistent state. You can also recover to an image copy that was taken with SHRLEVEL REFERENCE. This image copy serves as a point of consistency.

However, recovering objects to a quiesce point can be faster than recovering to any RBA or LRSN, because no work has to be backed out. Also, you might want to establish quiesce points for related sets of objects if you need to plan for point-in-time recovery for the entire set. For point-in-time recoveries, all objects in a table space set need to be recovered to the same point in time.

To obtain a common quiesce point for related table spaces, use the QUIESCE utility with the TABLESPACENSET option. For the purposes of the QUIESCE utility, a table space set includes the following sets for objects:

- A group of table spaces that have a referential relationship
- A base table space with all of its LOB table spaces
- A base table space with all of its XML table spaces
- A table space with a system-period temporal table and the table space with the related history table
- A table space that includes an archive-enabled table and the table space that contains the associated archive table

If you use QUIESCE TABLESPACE instead and do not include every member of the table space set, you might have problems when you run RECOVER on table spaces in the set. RECOVER checks if a complete table space set is recovered to a single point in time. If the complete table space set is not recovered to a single point in time, RECOVER places all dependent table spaces in CHECK-pending (CHKP) status.

When you use QUIESCE WRITE YES on a table space, the utility records the quiesce point in SYSIBM.SYSCOPY. QUIESCE inserts a SYSCOPY row that specifies ICTYPE='Q' for each related index that is defined with COPY=YES.

Related concepts:
“Point-in-time recovery” on page 495

Related reference:
“Syntax and options of the QUIESCE control statement” on page 416
Running QUIESCE on a table space in pending status

When you run QUIESCE on a table space in a pending status, the output will contain various messages.

If you run QUIESCE on a table space in COPY-pending, CHECK-pending, or RECOVER-pending status, it terminates with messages that are similar to those messages shown in the following figure.

```
DSNU0001  DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = R92341Q
DSNU0501  DSNUGUTC - QUIESCE TABLESPACE UTQPD22A.UTQPS22D
                    TABLESPACE UTQPD22A.UTQPS22E
                    TABLESPACE UTQPD22A.EMPPROJA
DSNU4711  $ DSNUQ0IA COPY PENDING ON TABLESPACE UTQPD22A.EMPPROJA PROHIBITS
                    PROCESSING
DSNU0121  DSNUGBAC - UTILITY EXECUTION TERMINATED, HIGHEST RETURN CODE=8
```

Figure 47. Termination messages when you run QUIESCE on a table space with pending restrictions

When you run QUIESCE on a table space or index space that is in COPY-pending, CHECK-pending, or RECOVER-pending status, you might also receive one or more of the messages that are shown in the following figure.

```
DSNU2021  csect RECOVER PENDING ON TABLESPACE... PROHIBITS PROCESSING
DSNU2031  csect RECOVER PENDING ON INDEX ... PROHIBITS PROCESSING
DSNU2041  csect PAGESET REBUILD PENDING ON INDEX ... PROHIBITS PROCESSING
DSNU2081  csect GROUP BUFFER POOL RECOVER PENDING ON INDEX ... PROHIBITS PROCESSING
DSNU2091  csect RESTART PENDING ON ... PROHIBITS PROCESSING
DSNU2111  csect CHECK PENDING ON ... PROHIBITS PROCESSING
DSNU2141  csect REBUILD PENDING ON INDEX ... PROHIBITS PROCESSING
DSNU2151  csect REFRESH PENDING ON ... PROHIBITS PROCESSING
DSNU4711  csect COPY PENDING ON TABLESPACE ... PROHIBITS PROCESSING
DSNU5681  csect INDEX ... IS IN INFORMATIONAL COPY PENDING
```

Figure 48. Messages for pending restrictions on QUIESCE

Reasons why QUIESCE fails to write to disk

The QUIESCE utility attempts to write pages of each table space to disk. Any of the following conditions can cause this write to fail:

- The table space has a write error range.
- The table space has deferred restart pending.
- An I/O error occurs.

If any of the preceding conditions occur, QUIESCE terminates with a return code of 4 and issues a DSNU473I warning message.

Related information:

- DSNU473I (Db2 Messages)

Termination and restart of QUIESCE

You can terminate and restart the QUIESCE utility.
If you use **TERM UTILITY** to terminate QUIESCE when it is active, QUIESCE releases the drain locks on table spaces. If QUIESCE is stopped, the drain locks have already been released.

You can restart a QUIESCE utility job, but it starts from the beginning again.

QUIESCE specifies whether the changed pages from the table spaces and index spaces are to be written to disk. The default option, YES establishes a quiesce point and writes the changed pages from the table spaces and index spaces to disk. The NO option establishes a quiesce point, but does not write the changed pages from the table spaces and index spaces to disk. QUIESCE is not performed on table spaces with the NOT LOGGED attribute.

**Related tasks:**

"Restarting an online utility” on page 44

---

**Sample QUIESCE control statements**

Use the sample control statements as models for developing your own QUIESCE control statements.

**Example 1: Establishing a quiesce point for three table spaces**

The following control statement specifies that the QUIESCE utility is to establish a quiesce point for table spaces DSN8D81A.DSN8S81D, DSN8D81A.DSN8S81E, and DSN8D81A.DSN8S81P.

```plain
//STEP1 EXEC DSNUPROC,UID='IUIQU2UD.QUIESC2',
    //   UTPROC='',SYSTEM='DSN'
    //SYSIN DD *
QUIESCE TABLESPACE DSN8D81A.DSN8S81D
       TABLESPACE DSN8D81A.DSN8S81E
       TABLESPACE DSN8D81A.DSN8S81P
/*
```

The following example shows the output that the preceding command produces.

```plaintext
DSNU0001 DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = TEMP
DSNU0441 DSNUGUTS - PROCESSING SYSIN AS EBCDIC
DSNU0501 DSNUGUTC - QUIESCE TABLESPACE DSN8D81A.DSN8S81D
       TABLESPACE DSN8D81A.DSN8S81E
       TABLESPACE DSN8D81A.DSN8S81P
DSNU4771 = DSNUQUIA - QUIESCE SUCCESSFUL FOR TABLESPACE DSN8D81A.DSN8S81D
DSNU4771 = DSNUQUIA - QUIESCE SUCCESSFUL FOR TABLESPACE DSN8D81A.DSN8S81E
DSNU4771 = DSNUQUIA - QUIESCE SUCCESSFUL FOR TABLESPACE DSN8D81A.DSN8S81P
DSNU4741 = DSNUQUIA - QUIESCE AT RBA 0000004E3B78 AND AT LRSN 0000004E3B78
DSNU4751 = DSNUQUIB - QUIESCE UTILITY COMPLETE, ELAPSED TIME= 00:00:02
DSNU0101 DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```

Figure 49. Example output from a QUIESCE job that establishes a quiesce point for three table spaces

**Example 2: Establishing a quiesce point for a list of objects**

In the following example, the QUIESCE control statement uses a list to specify that the QUIESCE utility is to establish a quiesce point for the same table spaces as in example 1. The list is defined in the LISTDEF utility control statement.

```plaintext
//STEP1 EXEC DSNUPROC,UID='IUIQU2UD.QUIESC2',
    //   UTPROC='',SYSTEM='DSN'
    //SYSIN DD *
    //DSNUPROC.SYSIN DD *
```

Chapter 24. QUIESCE  423
The following example shows the output that the preceding command produces.

Example 3: Establishing a quiesce point for a table space set.

The following control statement specifies that QUIESCE is to establish a quiesce point for the indicated table space set. In this example, the table space set includes table space DSN8D81A.DSN8S81D and all table spaces that are referentially related to it. Run REPORT TABLESPACESET to obtain a list of table spaces that are referentially related.

QUIESCE TABLESPACESET TABLESPACE DSN8D81A.DSN8S81D

The following example shows the output that the preceding command produces.

Example 4: Establishing a quiesce point without writing the changed pages to disk

In the following example, the control statement specifies that the QUIESCE utility is to establish a quiesce point for table space DSN8D81A.DSN8S81D, without writing the changed pages to disk. (The default is to write the changed pages to disk.) In this example, a quiesce point is established for COPY YES indexes, but not for COPY NO indexes. Note that QUIESCE jobs with the WRITE YES option, which is the default, process both COPY YES indexes and COPY NO indexes. For
both QUIESCE WRITE YES jobs and QUIESCE WRITE NO jobs, the utility inserts a row in SYSIBM.SYSCOPY for each COPY YES index.

```sql
//STEP1 EXEC DSNUPROC,UID='IUIQU2UD.QUIESC2',
// UTPROC='',SYSTEM='DSN'
//SYSIN DD *
//DSNUPROC.SYSIN DD *
QUIESCE TABLESPACE DSN8D81A.DSN8S81D WRITE NO
//--
```

The preceding command produces the output that is shown in the following example. Notice that the COPY YES index EMPNOI is placed in informational COPY-pending (ICOPY) status:

```
DSNU000I DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = TEMP
DSNU1044I DSNUGTIS - PROCESSING SYSIN AS EBCDIC
DSNU050I DSNUGUTC - QUIESCE TABLESPACE DSN8D81A.DSN8S81D WRITE NO
DSNU477I = DSNUQUIA - QUIESCE SUCCESSFUL FOR TABLESPACE DSN8D81A.DSN8S81D
DSNU477I = DSNUQUIA - QUIESCE SUCCESSFUL FOR INDEXSPACE DSN8D81A.EMPNOI
DSNU474I = DSNUQUIA - QUIESCE AT RBA 000004E892A3 AND AT LRSN 000004E892A3
DSNU568I = DSNUGRX - INDEX ADMF001.EMPNOI IS IN INFORMATIONAL COPY PENDING
DSNU475I = DSNUQUIB - QUIESCE UTILITY COMPLETE, ELAPSED TIME= 00:00:00
DSNU010I DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```

Figure 52. Example output from a QUIESCE job that establishes a quiesce point, without writing the changed pages to disk.

**Example 5: Establishing a quiesce point for a clone table space**

The following control statement specifies that the QUIESCE utility is to establish a quiesce point for the specified clone table space and its indexes, and write the changes to disk.

QUIESCE TABLESPACE DBJM0901.TPJM0901 WRITE YES CLONE
Chapter 25. REBUILD INDEX

The REBUILD INDEX online utility reconstructs indexes or index partitions from the table that they reference.

During the rebuild process, the REBUILD INDEX utility can also create a FlashCopy image copy of the indexes being rebuilt.

To avoid the cost of running the RUNSTATS utility afterward, you can also specify the STATISTICS option to collect inline statistics when you run the REBUILD INDEX utility.

You can use REBUILD INDEX in conjunction with z/OS DFSMS data set encryption to encrypt or decrypt indexes. REBUILD INDEX always allocates new Db2-managed data sets unless the REUSE option is specified.

Restriction: REBUILD INDEX SHRLEVEL CHANGE should only be used to fix a broken or restricted index, to build an index after DEFER, or to support or unencrypt previously encrypted data sets. You should not use the REBUILD INDEX SHRLEVEL CHANGE utility to move an index to different volumes; instead you should use the online REORG utility. REBUILD INDEX SHRLEVEL CHANGE on a unique index will not allow the INSERT option, the DELETE option, or updates that affect the unique index.

Authorization required

To execute this utility, you must use a privilege set that includes one of the following authorities:

- RECOVERDB privilege for the database
- STATS privilege for the database is required if the STATISTICS keyword is specified.
- DBADM or DBCTRL authority for the database. If the object on which the utility operates is in an implicitly created database, DBADM authority on the implicitly created database or DSNDB04 is required.
- DATAACCESS authority
- System DBADM authority
- SYSCTRL or SYSADM authority

If either the FLASHCOPY YES or FLASHCOPY CONSISTENT option is specified, the user ID that invokes the REBUILD INDEX utility must have the authority to execute the DFSMSdss COPY command.

To run REBUILD INDEX STATISTICS REPORT YES, you must use a privilege set that includes the SELECT privilege on the catalog tables.

Execution phases of REBUILD INDEX

The REBUILD INDEX utility operates in the following phases:

UTILINIT
  Performs initialization and setup.
UNLOAD
  Unloads index entries.

SORT  Sorts unloaded index entries.

BUILD  Builds indexes.

SORTBLD  Sorts and builds a table space for parallel index build processing.

UTILTERM  Performs cleanup.

Related tasks:
  - Collecting statistics by using Db2 utilities (Db2 Performance)
  - Reducing the cost of collecting statistics (Db2 Performance)

Syntax and options of the REBUILD INDEX control statement

The REBUILD INDEX utility control statement, with its multiple options, defines the function that the utility job performs.

You can create a control statement with the ISPF/PDF edit function. After creating it, save it in a sequential or partitioned data set. When you create the JCL for running the job, use the SYSIN DD statement to specify the name of the data set that contains the utility control statement.
Syntax diagram

Notes:
1 All listed indexes must reside in the same table space.

table-space-spec:

CHP | 25. REBUILD INDEX | 429
Notes:
1  The KEYCARD option is deprecated. The functionality previously controlled by KEYCARD is incorporated into the default processing of inline statistics during the execution of REBUILD INDEX STATISTICS and cannot be disabled.

Option descriptions

INDEX creator-id.index-name
Indicates the qualified name of the index to be rebuilt. Use the form creator-id.index-name to specify the name.

creator-id
Specifies the creator of the index. This qualifier is optional. If you omit the qualifier creator-id, Db2 uses the user identifier for the utility job.

index-name
Specifies the qualified name of the index that is to be rebuilt. For an index, you can specify either an index name or an index space name. Enclose the index name in quotation marks if the name contains a blank.

To rebuild multiple indexes, separate each index name with a comma. All listed indexes must reside in the same table space. If more than one index is listed and the TABLESPACE keyword is not specified, Db2 locates the first valid index name that is cited and determines the table space in which that index resides. That table space is used as the target table space for all other valid index names that are listed.

INDEXSPACE database-name.index-space-name
Specifies the qualified name of the index space that is obtained from the SYSIBM.SYSINDEXES table.

database-name
Specifies the name of the database that is associated with the index. This qualifier is optional.

index-space-name
Specifies the qualified name of the index space to copy. For an index, you can specify either an index name or an index space name.

If you specify more than one index space, they must all be defined on the same table space.
For an index, you can specify either an index name or an index space name.

(ALL)
Specifies that all indexes in the table space that is referred to by the TABLESPACE keyword are to be rebuilt. If you specify ALL, only indexes on the base table are included.

TABLESPACE database-name.table-space-name
Specifies the table space from which all indexes are to be rebuilt.

database-name
Identifies the database to which the table space belongs.

The default value is DSND04.

table-space-name
Identifies the table space from which all indexes are to be rebuilt.

PART integer
Specifies the physical partition of a partitioning index or a data-partitioned secondary index in a partitioned table that is to be rebuilt. When the target of the REBUILD operation is a nonpartitioned secondary index, the utility reconstructs logical partitions. If any of the following situations are true for a nonpartitioned index, you cannot rebuild individual logical partitions:
• the index was created with DEFER YES
• the index must be completely rebuilt (This situation is likely in a disaster recovery scenario)
• the index is in page set REBUILD-pending (PSRBD) status
For these cases, you must rebuild the entire index.

integer is the physical partition number. It must be in the range from 1 to the number of partitions that are defined for the table space. The maximum is 4096.

You cannot specify PART with the LIST keyword. Use LISTDEF PARTLEVEL for partitioning or data-partitioned secondary indexes. The PARTLEVEL keyword is ignored for nonpartitioned secondary indexes; if it is specified, the entire index will be rebuilt.

LIST listdef-name
Specifies the name of a previously defined LISTDEF list name. The utility allows one LIST keyword for each REBUILD INDEX control statement. The list must contain either all index spaces or all table spaces. For a table space list, REBUILD is invoked once per table space. For an index space list, Db2 groups indexes by their related table space and executes the rebuild once per table space. This utility will only process clone data if the CLONE keyword is specified. The use of CLONED YES on the LISTDEF utility control statement is not sufficient.

The partitions or partition ranges for partitioning or data-partitioned secondary indexes can be specified in a list. Logical partitions for nonpartitioned secondary indexes cannot be specified in a list.

SHRLEVEL
Indicates the type of access that is to be allowed for the index, table space, or partition that is to be checked during REBUILD INDEX processing.

REFERENCE
Specifies that applications can read from but cannot write to the table
space or partition that REBUILD accesses. Applications cannot read or write from the index REBUILD is building.

**CHANGE**

Specifies that applications can read from and write to the table space or partition. The index is placed in RBDP and can be avoided by dynamic SQL. CHANGE is invalid for indexes over XML tables.

Do not specify SHRLEVEL CHANGE for an index on a NOT LOGGED table space.

**Restriction:**

- SHRLEVEL CHANGE is not well suited for unique indexes and concurrent DML because the index is placed in RBDP while being built. Inserts and updates of the index will fail with a resource unavailable (-904) because uniqueness checking cannot be done while the index is in RBDP.
- SHRLEVEL CHANGE is not allowed on not logged tables, XML indexes, or spatial indexes.

**MAXRO**

Specifies the maximum amount of time for the last iteration of log processing. During that iteration, applications have read-only access.

The actual execution time of the last iteration might exceed the specified value for MAXRO.

integer

integer is the number of seconds. Specifying a small positive value reduces the length of the period of read-only access, but it might increase the elapsed time for REBUILD INDEX to complete. If you specify a huge positive value, the second iteration of log processing is probably the last iteration.

The default value is the value of the lock timeout system parameter **IRLMRWT**.

**LONGLOG**

Specifies the action that Db2 is to perform, after sending a message to the console, if the number of records that the next iteration of logging is to process is not sufficiently lower than the number that the previous iterations processed. This situation means that the reading of the log by the REBUILD INDEX utility is not being done at the same time as the writing of the application log.

**CONTINUE**

Specifies that until the time on the JOB statement expires, Db2 is to continue performing reorganization, including iterations of log processing, if the estimated time to perform an iteration exceeds the time that is specified for MAXRO.

**TERM**

Specifies that Db2 is to terminate the reorganization after the delay that is specified by the DELAY parameter.

**DRAIN**

Specifies that Db2 is to drain the write claim class after the delay that is specified by the DELAY parameter. This action forces the final iteration of log processing to occur.

**DELAY integer**

Specifies the minimum integer between the time that REBUILD send the
LONGLOG message to the console and the time that REBUILD performs the action the LONGLOG parameter specifies.

The integer specifies the number of seconds.

The default value is **1200**.

**DRAIN_WAIT**

Specifies the number of seconds that REBUILD INDEX is to wait when draining the table space or index. The specified time is the aggregate time for objects that are to be checked. This value overrides the values that are specified by the IRLMRWT and UTIMOUT subsystem parameters.

integer can be any integer from 0 to 1800. If you do not specify DRAIN_WAIT or specify a value of 0, the utility uses the value of the lock timeout subsystem parameter IRLMRWT.

**RETRY integer**

Specifies the maximum number of retries that REBUILD INDEX is to attempt.

integer can be any integer from 0 to 255. If you do not specify RETRY, REBUILD INDEX uses the value of the utility multiplier system parameter UTIMOUT.

Specifying RETRY can increase processing costs and result in multiple or extended periods during which the specified index, table space, or partition is in read-only access.

**RETRY_DELAY integer**

Specifies the minimum duration, in seconds, between retries. integer can be any integer from 1 to 1800.

If you do not specify RETRY_DELAY, REBUILD INDEX uses the DRAIN_WAIT value × RETRY value.

**CLONE**

Indicates that REBUILD INDEX is to reconstruct only the specified indexes that are on clone tables. This utility will only process clone data if the CLONE keyword is specified. The use of CLONED YES on the LISTDEF statement is not sufficient. If you specify CLONE, you cannot specify STATISTICS. Statistics are not collected for clone objects.

**SCOPE**

Indicates the scope of the rebuild organization of the specified index or indexes.

- **ALL** Indicates that you want the specified index or indexes to be rebuilt.
- **PENDING** Indicates that you want the specified index or indexes with one or more partitions in REBUILD-pending (RBDP), REBUILD-pending star (RBDP*), page set REBUILD-pending (PSRBD), RECOVER-pending (RECP), or advisory REORG-pending (AREO*) state to be rebuilt.

**REUSE**

Specifies that REBUILD should logically reset and reuse Db2-managed data sets without deleting and redefining them. If you do not specify REUSE, Db2 deletes and redefines Db2-managed data sets to reset them.

If you are rebuilding the index because of a media failure, do not specify REUSE.
If a data set has multiple extents, the extents are not released if you use the REUSE parameter.

SORTDEV device-type

Specifies the device type for temporary data sets that are to be dynamically allocated by the sort program. For device-type, you can specify any disk device that is valid on the DYNALLOC parameter of the SORT or OPTION options for the sort program. Tape devices are not supported by the sort program.

device-type is the device type.

A TEMPLATE specification does not dynamically allocate sort work data sets. The SORTDEV keyword controls dynamic allocation of these data sets.

SORTNUM integer

Specifies the number of temporary data sets that are to be dynamically allocated by the sort program. If you omit SORTDEV, SORTNUM is ignored. If you use SORTDEV and omit SORTNUM, no value is passed to the sort program; the sort program uses its own default.

integer is the number of temporary data sets that can range from 2 to 255.

You need at least two sort work data sets for each sort. The SORTNUM value applies to each sort invocation in the utility. For example, if there are three indexes, SORTKEYS is specified, there are no constraints limiting parallelism, and SORTNUM is specified as 8, then a total of 24 sort work data sets will be allocated for a job.

Each sort work data set consumes both above the line and below the line virtual storage, so if you specify too high a value for SORTNUM, the utility may decrease the degree of parallelism due to virtual storage constraints, and possibly decreasing the degree down to one, meaning no parallelism.

Important: The SORTNUM keyword is ignored if the UTSORTAL subsystem parameter is set to YES and the IGNSORTN subsystem parameter is set to YES.

STATISTICS

Specifies that index statistics are to be collected.

If you specify the STATISTICS and UPDATE options, statistics are stored in the Db2 catalog. You cannot collect inline statistics for indexes on the catalog and directory tables.

Restriction:

- If you specify STATISTICS for encrypted data, Db2 might not provide useful statistics on this data.
- You cannot specify STATISTICS for a clone index.

REPORT

Indicates whether a set of messages to report the collected statistics is to be generated.

NO Indicates that the set of messages is not to be sent as output to SYSPRINT.

YES Indicates that the set of messages is to be sent as output to SYSPRINT. The generated messages are dependent on the combination of keywords (such as TABLESPACE, INDEX, TABLE, and COLUMN) that you specify with the RUNSTATS utility. However, these messages are not dependent on the specification of the UPDATE option. REPORT YES always generates a report of SPACE and ACCESSPATH statistics.
KEYCARD

The KEYCARD option is deprecated in the utility control statement and no longer needs to be specified to collect cardinality statistics on the values in the key columns of an index.

When the STATISTICS option is specified, the utility automatically collects all of the distinct values in all of the 1 to \( n \) key column combinations for the indexes being rebuilt. \( n \) is the number of columns in the index. With the deprecation of KEYCARD, this functionality cannot be disabled.

The utility tolerates the specification of the KEYCARD option. The utility does not issue any messages if the control statement includes or excludes the KEYCARD option when STATISTICS is specified.

FREQVAL

Controls the collection of frequent-value statistics. If you specify FREQVAL, it must be followed by the NUMCOLS keyword.

NUMCOLS

Indicates the number of key columns that are to be concatenated when collecting frequent values from the specified index. If you specify 3, the utility collects frequent values on the concatenation of the first three key columns.

The default value is 1, which means that Db2 is to collect frequent values only on the first key column of the index.

COUNT

Indicates the number of frequent values that are to be collected. If you specify 15, the utility collects 15 frequent values from the specified key columns.

The default value is 10.

HISTOGRAM

Indicates that histogram statistics are requested for the specified index.

NUMCOLS

The number of key columns that are to be concatenated when collecting histogram statistics from the specified index.

NUMQUANTILES

The integer values that follow NUMQUANTILES indicates the number quantiles are requested. The integer value must be greater than or equal to 1.

Histogram statistics can be collected only on keys with the same order. If the specified key columns for histogram statistics are of mixed or random order, a DSNU633I warning message is issued.

Histogram statistics that you collect through REBUILD INDEX are not the same as histogram statistics that you collect through RUNSTATS. Histogram statistics that you collect with REBUILD INDEX are only rough estimates. To obtain more exact statistics, use RUNSTATS.

Related information:

UPDATE

Indicates whether the collected statistics are to be inserted into the catalog
tables. **UPDATE** also allows you to select statistics that are used for access path selection or statistics that are used by database administrators.

**ALL**
Indicates that all collected statistics are to be updated in the catalog.

**ACCESSPATH**
Indicates that the only catalog table columns that are to be updated are those that provide statistics that are used for access path selection.

**SPACE**
Indicates that the only catalog table columns that are to be updated are those that provide statistics to help the database administrator assess the status of a particular table space or index.

**NONE**
Indicates that catalog tables are not to be updated with the collected statistics. This option is valid only when **REPORT YES** is specified.

**HISTORY**
Records all catalog table inserts or updates to the catalog history tables.

The default is supplied by the value that is specified in **STATISTICS HISTORY** on panel **DSNTIP6**.

**ALL**
Indicates that all collected statistics are to be updated in the catalog history tables.

**ACCESSPATH**
Indicates that the only catalog history table columns that are to be updated are those that provide statistics that are used for access path selection.

**SPACE**
Indicates that only space-related catalog statistics are to be updated in catalog history tables.

**NONE**
Indicates that catalog history tables are not to be updated with the collected statistics.

**FORCEROLLUP**
Specifies whether aggregation or rollup of statistics is to take place when you execute **RUNSTATS** even if some indexes or index partitions are empty. This keyword enables the optimizer to select the best access path.

The following options are available for the **FORCEROLLUP** keyword:

**YES**
Indicates that forced aggregation or rollup processing is to be done, even though some indexes or index partitions might not contain data.

**NO**
Indicates that aggregation or rollup is to be done only if data is available for all indexes or index partitions.

If data is not available, the utility issues **DSNU623I** message if you have set the installation value for **STATISTICS ROLLUP** on panel **DSNTIP6** to **NO**.

**FLASHCOPY**
Specifies whether FlashCopy technology is used to create a copy of the object. Valid values are **YES**, **NO**, or **CONSISTENT**. When FlashCopy is used, a separate data set is created for each partition or piece of the object.
The FlashCopy specifications on the utility control statement override any specifications for FlashCopy that are defined by using the Db2 subsystem parameters. If the FlashCopy subsystem parameters specify the use of FlashCopy as the default behavior of this utility, the FLASHCOPY option can be omitted from the utility control statement.

**Important:** If the input data set is less than one cylinder, FlashCopy technology might not be used for copying the objects regardless of the FLASHCOPY settings. The copy is performed by IDCAMS if FlashCopy is not used.

**NO** Specifies that no FlashCopy is made. NO is the default value for FLASHCOPY.

**YES** Specifies that FlashCopy technology is used to copy the object.

Specify YES only if the Db2 data sets are on FlashCopy Version 2 disk volumes.

**Important:** Under the following circumstances, the REBUILD INDEX utility might not use FlashCopy even though YES is specified:

- FlashCopy Version 2 disk volumes are not available
- The source tracks are already the target of a FlashCopy operation
- The target tracks are the source of a FlashCopy operation
- The maximum number of relationships for the copy is exceeded

In the event that FlashCopy is not used, the REBUILD INDEX utility uses traditional I/O methods to copy the object, which can result in longer than expected execution time.

**CONSISTENT**

Specifies that FlashCopy technology is used to copy the object. Because the copies created by the REBUILD INDEX utility are already consistent, the utility treats a specification of CONSISTENT the same as a specification of YES.

**Related information:**

- "FlashCopy image copies" on page 157
- "Subsystem parameters for refining DFSMSdss COPY operation with utilities" on page 40

**PARALLEL num-subtasks**

Specifies the maximum number of subtasks that are to be started in parallel to rebuild indexes. If the PARALLEL keyword is omitted, the maximum number of subtasks is limited by either the number of partitions being unloaded or the number of indexes built.

REBUILD INDEX typically allocates subtasks in groups of two or three, so the actual number of subtasks that are started might be less than the number specified on PARALLEL.

The specified number of subtasks for PARALLEL always overrides the specification of the PARAMDEG_UTIL subsystem parameter, so PARALLEL can be smaller or larger than the value of PARAMDEG_UTIL.

**num-subtasks**

Specifies the maximum number of subtasks and must be an integer between 0 and 32767, inclusive. If the specified value for num-subtasks is
greater than 32767, the REBUILD INDEX statement fails. If 0 or no value is
specified for num-subtasks, the REBUILD INDEX utility uses the optimal
number of parallel subtasks. If the specified value for num-subtasks is
greater than the calculated optimal number, the REBUILD INDEX utility
limits the number of parallel subtasks to the optimal number with applied
constraints.

RBALRSN_CONVERSION
Specifies the RBA or LRSN format of the target object after the completion of
the REBUILD INDEX utility. If the keyword is not specified, the conversion
specified in the UTILITY_OBJECT_CONVERSION subsystem parameter is
accepted.

NONE
Specifies that no conversion is performed.

The utility fails if RBALRSN_CONVERSION NONE is specified on a table
space that is in basic 6-byte format and the
UTILITY_OBJECT_CONVERSION subsystem parameter is set to
NOBASIC.

BASIC
Specifies that if an object is found in extended 10-byte format, it is
converted to 6-byte basic format.

The utility fails if RBALRSN_CONVERSION BASIC is specified and the
UTILITY_OBJECT_CONVERSION subsystem parameter is set to
NOBASIC.

EXTENDED
Specifies that if an object is found in basic 6-byte format, it is converted to
10-byte extended format.

If RBALRSN_CONVERSION is not specified, and subsystem parameter
UTILITY_OBJECT_CONVERSION is set to BASIC or EXTENDED,
RBALRSN_CONVERSION defaults to the UTILITY_OBJECT_CONVERSION
value. If RBALRSN_CONVERSION is not specified, and subsystem parameter
UTILITY_OBJECT_CONVERSION is set to NOBASIC,
RBALRSN_CONVERSION defaults to EXTENDED.

If a CLONE relationship exists, the page set conversion cannot be performed.
For clone relationships, you must drop the clone table, convert the base table
to extended 10-byte format, and then re-create the clone table.

REBUILD of a node ID index, when converting the page format to extended,
does not convert versioned XML table spaces that are associated with that base
table space.

Specify RBALRSN_CONVERSION NONE during Disaster Recovery scenarios
to avoid page set format conversions, which would complicate the recovery,
especially when you rebuild indexes over the catalog and directory table
spaces. Alternatively, set the UTILITY_OBJECT_CONVERSION subsystem
parameter to NONE until the Disaster Recovery completes.

FCCOPYDDN
Specifies the template to be used to create the FlashCopy image copy data set
names. If a value is not specified for FCCOPYDDN on the REBUILD INDEX
control statement when FlashCopy is used, the value specified on the
FCCOPYDDN subsystem parameter determines the template to be used.
The data set names for the FlashCopy image copy are allocated according to the template specification. For table space or index space level FlashCopy image copies, because a data set is allocated for each partition or piece, ensure that the data set naming convention in the template specification is unique enough. Use the &DSNUM variable, which resolves to a partition number or piece number at execution time.

Related reference:

- [Db2 Sort for z/OS](#)

Related information:

- [DFSORT Application Programming Guide](#)

Before running REBUILD INDEX

Certain activities might be required before you run the REBUILD INDEX utility, depending on your situation.

Because the data that Db2 needs to build an index is in the table space on which the index is based, you do not need image copies of indexes. To rebuild the index, you do not need to recover the table space, unless it is also damaged. You do not need to rebuild an index merely because you have recovered the table space on which it is based.

If you recover a table space to a prior point in time and do not recover all the indexes to the same point in time, you must rebuild all of the indexes.

Some logging might occur if both of the following conditions are true:

- The index is a nonpartitioning index.
- The index is being concurrently accessed either by SQL on a different partition of the same table space or by a utility that is run on a different partition of the same table space.

Running REBUILD INDEX when the index has a VARBINARY column.

If you run REBUILD INDEX against an index with the following characteristics, REBUILD INDEX fails:

- The index was created on a VARBINARY column or a column with a distinct type that is based on a VARBINARY data type.
- The index column has the DESC attribute.

To fix the problem, alter the column data type to BINARY, and then run REBUILD INDEX.

Data sets that REBUILD INDEX uses

The REBUILD INDEX utility uses a number of data sets during its operation.

The following table lists the data sets that REBUILD INDEX uses. The table lists the DD name that is used to identify the data set, a description of the data set, and an indication of whether it is required. Include statements in your JCL for each required data set and any optional data sets that you want to use.
Table 60. Data sets that REBUILD INDEX uses

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSIN</td>
<td>Input data set that contains the utility control statement.</td>
<td>Yes</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>Output data set for messages.</td>
<td>Yes</td>
</tr>
<tr>
<td>STPRIN01</td>
<td>A data set that contains messages from the sort program (usually, SYSOUT or DUMMY). This data set is used when statistics are collected on at least one data-partitioned secondary index.</td>
<td>No</td>
</tr>
<tr>
<td>Work data sets</td>
<td>Temporary data sets for sort input and output when sorting keys. If index build parallelism is used, the DD names have the form SWnnWKmmmm. If index build parallelism is not used, the DD names have the form SORTWKnn.</td>
<td>Yes</td>
</tr>
<tr>
<td>Sort work data sets</td>
<td>Temporary data sets for sort input and output when collecting inline statistics on at least one data-partitioned secondary index. The DD names have the form ST01WKnn.</td>
<td>No</td>
</tr>
<tr>
<td>UTPRINT</td>
<td>A data set that contains messages from the sort program (usually, SYSOUT or DUMMY).</td>
<td>Yes</td>
</tr>
<tr>
<td>FlashCopy image copy data sets</td>
<td>For copies of the entire index space, a separate VSAM data set for each partition or piece that is contained in the index space. For partition-level or piece-level copies, a VSAM data set for each partition or piece that is being copied.</td>
<td>No</td>
</tr>
</tbody>
</table>

Note:
1. STPRIN01 is required if statistics are being collected on at least one data-partitioned secondary index, but REBUILD INDEX dynamically allocates the STPRIN01 data set if UTPRINT is allocated to SYSOUT.
2. Required when collecting inline statistics on at least one data-partitioned secondary index.
3. If the DYNALLOC parm of the SORT program is not turned on, you need to allocate the data set. Otherwise, the sort program dynamically allocates the temporary data set.
4. It is recommended that you use dynamic allocation by specifying SORDEVT in the utility statement because dynamic allocation reduces the maintenance required of the utility job JCL.
5. Required if you specify the FLASHCOPY YES.

The following object is named in the utility control statement and does not require a DD statement in the JCL:

Table space
Object whose indexes are to be rebuilt.

Calculating the size of the SORTWKnn data set

To calculate the approximate size (in bytes) of the SORTWKnn data set, use the following formula:

\[ 2 \times (longest\ index\ key + c) \times (number\ of\ extracted\ keys) \]
**longest index key**

The length of the longest index key that is to be processed by the subtask.

If the index is of varying length, the longest key is the maximum possible length of a key with all varying-length columns that are padded to their maximum length, plus 2 bytes for each varying-length column in the index. For example, if an index with 3 columns (A, B, and C) has length values of CHAR(8) for A, VARCHAR(128) for B, and VARCHAR(50) for C, the longest key is calculated as follows:

\[ 8 + 128 + 50 + 2 + 2 = 190 \]

\( c \) A value as follows:

- 10 if the indexes that are being rebuilt are a mix of data-partitioned secondary indexes and nonpartitioned indexes
- 8 if the indexes that are being rebuilt are partitioned, or if none of them are data-partitioned secondary indexes.

**number of keys**

The number of keys from all indexes that the subtask sorts and processes.

Using two or three large SORTWKnn data sets are preferable to several small ones.

### Calculating the size of the ST01WKnn data set

To calculate the approximate size (in bytes) of the ST01WKnn data set, use the following formula:

\[ 2 \times (\text{maximum record length} \times \text{numcols} \times (\text{count} + 2) \times \text{number of indexes}) \]

The variables in the preceding formula have the following values:

**maximum record length**

Maximum record length of the SYSCOLDISTSTATS record that is processed when collecting frequency statistics (You can obtain this value from the RECLength column in SYSTABLES.)

**numcols**

Number of key columns to concatenate when you collect frequent values from the specified index.

**count**

Number of frequent values that Db2 is to collect.

Sort work data sets cannot span volumes. Smaller volumes require more sort work data sets to sort the same amount of data; therefore, large volume sizes can reduce the number of needed sort work data sets. When you allocate sort work data sets on disk, the recommended amount of space to allow provides at least 1.2 times the amount of data that is to be sorted.

Tape devices are not supported for sort work data sets.

**Related concepts:**

- "Data sets that online utilities use" on page 16

**Related reference:**

- [Db2 Sort for z/OS](#)

**Related information:**

- [DFSORT Application Programming Guide](#)
Concurrenty and compatibility for REBUILD INDEX

The REBUILD INDEX utility has certain concurrency and compatibility characteristics associated with it.

Db2 treats individual data and index partitions as distinct target objects. Utilities that operate on different partitions of the same table space or index space are compatible.

REBUILD INDEX SHRLEVEL CHANGE jobs cannot be run to rebuild indexes on the same table space concurrently. As an alternative, REBUILD INDEX can build indexes in parallel by specifying multiple indexes in a single utility statement. Concurrency for rebuilding indexes in different table space is still allowed, as is the concurrency in rebuilding different partitions of an index in a partitioned table space.

Restriction: REBUILD INDEX SHRLEVEL CHANGE should only be used to fix a broken or restricted index, or to build an index after DEFER. You should not use the REBUILD INDEX SHRLEVEL CHANGE utility to move an index to different volumes; instead you should use the online REORG utility. REBUILD INDEX SHRLEVEL CHANGE on a unique index will not allow the INSERT option, the DELETE option, or updates that affect the unique index.

Claims

The following table shows which claim classes REBUILD INDEX drains and any restrictive state that the utility sets on the target object.

Table 61. Claim classes of REBUILD INDEX operations.

<table>
<thead>
<tr>
<th>Target</th>
<th>REBUILD INDEX SHRLEVEL REFERENCE</th>
<th>REBUILD INDEX PART SHRLEVEL REFERENCE</th>
<th>REBUILD INDEX SHRLEVEL CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table space or partition</td>
<td>DW/UTRO</td>
<td>DW/UTRO</td>
<td>CR/UTRW</td>
</tr>
<tr>
<td>Partitioning index, data-partitioned secondary index, or physical partition</td>
<td>DA/UTUT</td>
<td>DA/UTUT</td>
<td>CR/UTRW</td>
</tr>
<tr>
<td>Nonpartitioned secondary index</td>
<td>DA/UTUT</td>
<td>DR</td>
<td>CR/UTRW</td>
</tr>
<tr>
<td>Logical partition of an index</td>
<td>N/A</td>
<td>DA/UTUT</td>
<td>CR/UTRW</td>
</tr>
</tbody>
</table>

Legend:
- CR - Claim the read claim class
- DA - Drain all claim classes; no concurrent SQL access
- DW - Drain the write claim class; concurrent access for SQL readers
- DR - Drains the repeatable-read claim class
- N/A - Not applicable
- UTUT - Utility restrictive state; exclusive control
- UTRO - Utility restrictive state; read-only access allowed
- UTRW - Utility restrictive state; read and write access allowed

Note:
1. Includes document ID indexes and node ID indexes over partitioned XML table spaces
2. Includes document ID indexes and node ID indexes over nonpartitioned XML table spaces and XML indexes
3. Includes logical partitions of an XML index over partitioned XML table spaces
Compatibility

The following table shows which utilities can run concurrently with REBUILD INDEX on the same target object. The target object can be an index space or a partition of an index space. If compatibility depends on particular options of a utility, that information is also shown. REBUILD INDEX does not set a utility restrictive state if the target object is DSND01.SYSUTILX.

Table 62. Compatibility of REBUILD INDEX with other utilities

<table>
<thead>
<tr>
<th>Action</th>
<th>REBUILD INDEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHECK DATA</td>
<td>No</td>
</tr>
<tr>
<td>CHECK INDEX</td>
<td>No</td>
</tr>
<tr>
<td>CHECK LOB</td>
<td>Yes</td>
</tr>
<tr>
<td>COPY INDEX</td>
<td>No</td>
</tr>
<tr>
<td>COPY TABLESPACE SHRLEVEL CHANGE</td>
<td>No</td>
</tr>
<tr>
<td>COPY TABLESPACE SHRLEVEL REFERENCE</td>
<td>Yes</td>
</tr>
<tr>
<td>DIAGNOSE</td>
<td>Yes</td>
</tr>
<tr>
<td>LOAD</td>
<td>No</td>
</tr>
<tr>
<td>MERGECOPY</td>
<td>Yes</td>
</tr>
<tr>
<td>MODIFY</td>
<td>Yes</td>
</tr>
<tr>
<td>QUIESCE</td>
<td>No</td>
</tr>
<tr>
<td>REBUILD INDEX</td>
<td>No</td>
</tr>
<tr>
<td>RECOVER INDEX</td>
<td>No</td>
</tr>
<tr>
<td>RECOVER TABLESPACE</td>
<td>No</td>
</tr>
<tr>
<td>REORG INDEX</td>
<td>No</td>
</tr>
<tr>
<td>REORG TABLESPACE UNLOAD CONTINUE or PAUSE</td>
<td>No</td>
</tr>
<tr>
<td>REORG TABLESPACE UNLOAD ONLY or EXTERNAL with cluster index</td>
<td>No</td>
</tr>
<tr>
<td>REORG TABLESPACE UNLOAD ONLY or EXTERNAL without cluster index</td>
<td>Yes</td>
</tr>
<tr>
<td>REPAIR LOCATE by KEY</td>
<td>No</td>
</tr>
<tr>
<td>REPAIR LOCATE by RID DELETE or REPLACE</td>
<td>No</td>
</tr>
<tr>
<td>REPAIR LOCATE by RID DUMP or VERIFY</td>
<td>Yes</td>
</tr>
<tr>
<td>REPAIR LOCATE INDEX PAGE DUMP or VERIFY</td>
<td>No</td>
</tr>
<tr>
<td>REPAIR LOCATE TABLESPACE or INDEX PAGE REPLACE</td>
<td>No</td>
</tr>
<tr>
<td>REPAIR LOCATE TABLESPACE PAGE DUMP or VERIFY</td>
<td>Yes</td>
</tr>
<tr>
<td>REPORT</td>
<td>Yes</td>
</tr>
<tr>
<td>RUNSTATS INDEX</td>
<td>No</td>
</tr>
<tr>
<td>RUNSTATS TABLESPACE</td>
<td>Yes</td>
</tr>
<tr>
<td>STOSPACE</td>
<td>Yes</td>
</tr>
<tr>
<td>UNLOAD</td>
<td>Yes</td>
</tr>
</tbody>
</table>

To run REBUILD INDEX on SYSIBM.DSNLUX01 or SYSIBM.DSNLUX02, ensure that REBUILD INDEX is the only utility in the job step and the only utility that is running in the Db2 subsystem. Unloading a base table that has LOB columns is not compatible with REBUILD INDEX.
Access with REBUILD INDEX SHRLEVEL

You can specify the level of access that you have to your data when running the REBUILD INDEX utility by using the SHRLEVEL option of REBUILD INDEX.

Before target indexes are built, they are first drained (DRAIN ALL), then placed in RBDP. The indexes are shown in UTRW states.

For rebuilding an index or a partition of an index, the SHRLEVEL option lets you choose the data access level that you have during the rebuild:

Log processing with SHRLEVEL CHANGE

When you specify SHRLEVEL CHANGE, Db2 processes the log. This step executes iteratively. The first iteration processes the log records that accumulated during the previous iteration. The iterations continue until one of these conditions is met:

- Db2 estimates that the time to perform the log processing in the next iteration will be less than or equal to the time that is specified by MAXRO. If this condition is met, the next iteration is the last.
- The number of log records that the next iteration will process is not sufficiently lower than the number of log records that were processed in the previous iteration. If this condition is met but the first two conditions are not, Db2 sends message DSNU377I to the console. Db2 continues log processing for the length of time that is specified by DELAY and then performs the action specified by LONGLOG.

Operator actions

LONGLOG specifies the action that Db2 is to perform if log processing is not occurring quickly enough. If the operator does not respond to the console message DSNU377I, the LONGLOG option automatically goes into effect. You can take one of the following actions:

- Execute the TERM UTILITY command to terminate the rebuild process.

Db2 does not take the action specified in the LONGLOG phrase if any one of these events occurs before the delay expires:

- A TERM UTILITY command is issued.
- Db2 estimates that the time to perform the next iteration is likely to be less than or equal to the time specified on the MAXRO keyword.
- REBUILD terminates for any reason (including the deadline).

Rebuilding index partitions

The REBUILD INDEX utility can rebuild one or more partitions of a partitioned index by extracting the keys from the data rows of the table on which they are based.

When you specify the PART option, one or more partitions from a partitioning index or a data-partitioned secondary index can be rebuilt. However, for nonpartitioned indexes, you cannot rebuild individual logical partitions in certain situations.

If any of the following situations are true for a nonpartitioned index, you cannot rebuild individual logical partitions:
- the index was created with DEFER YES
- the index must be completely rebuilt (This situation is likely in a disaster recovery scenario)
- the index is in page set REBUILD-pending (PSRBD) status

For these cases, you must rebuild the entire index.

**Rebuilding indexes on partition-by-growth table spaces**

The REBUILD INDEX Utility might reset more partitions than it repopulates. Any excess partitions will be empty after the REBUILD process.

**How to improve performance when rebuilding index partitions**

You can improve the performance of the REBUILD INDEX utility by taking certain actions.

If you use the PART option to rebuild only a single partition of an index, the utility does not need to scan the entire table space.

To rebuild several indexes (including data-partitioned secondary indexes) at the same time and reduce recovery time, use parallel index rebuild, or submit multiple index jobs.

When rebuilding nonpartitioned secondary indexes and partitions of partitioned indexes, this type of parallel processing on the same table space decreases the size of the sort data set, as well as the total time that is required to sort all the keys.

When you run the REBUILD INDEX utility concurrently on separate partitions of a partitioned index (either partitioning or secondary), the sum of the processor time is approximately the time for a single REBUILD INDEX job to run against the entire index. For partitioning indexes, the elapsed time for running concurrent REBUILD INDEX jobs is a fraction of the elapsed time for running a single REBUILD INDEX job against an entire index.

**When to use SHRLEVEL CHANGE:**

Schedule REBUILD with SHRLEVEL CHANGE when the rate of writing is low and transactions are short. Avoid scheduling REBUILD with SHRLEVEL CHANGE when low-tolerance applications are executing.

**When to use DRAIN_WAIT:**

The DRAIN_WAIT option provides improved control over the time online REBUILD waits for drains. Also, because the DRAIN_WAIT is the aggregate time that online REBUILD is to wait to perform a drain on a table space and associated indexes, the length of drains is more predictable than it is when each partition and index has its own individual waiting-time limit.

By specifying a short delay time (less than the system timeout value, IRLMRWT), you can reduce the impact on applications by reducing timeouts. You can use the RETRY option to give opportunities for the online REBUILD INDEX utility to complete successfully. If you do not want to use RETRY processing, you can still use DRAIN_WAIT to set a specific and more consistent limit on the length of drains.
RETRY allows an online REBUILD that is unable to drain the objects that it requires to try again after a set period (RETRY_DELAY). Objects will remain in their original state if the drain fails in the LOG phase.

Because application SQL statements can queue behind any unsuccessful drain that the online REBUILD has tried, define a reasonable delay before you retry to allow this work to complete; the default is lock timeout subsystem parameter IRLMRWT.

When the default DRAIN WRITERS is used with SHRLEVEL CHANGE and RETRY, multiple read-only log iterations can occur. Because online REBUILD can have to do more work when RETRY is specified, multiple or extended periods of restricted access might occur. Applications that run with REBUILD must perform frequent commits. During the interval between retries, the utility is still active; consequently, other utility activity against the table space and indexes is restricted.

**Recommendation:** Run online REBUILD during light periods of activity on the table space or index.

**Related concepts:**
“Rebuilding multiple indexes”

---

**Rebuilding multiple indexes**

When you process both node ID indexes and XML indexes together, they are processed sequentially. First the node ID index is processed and then the XML index.

**Building indexes in parallel**

Parallel index build reduces the elapsed time for a REBUILD INDEX job by sorting the index keys and rebuilding multiple indexes or index partitions in parallel, rather than sequentially. Optimally, a pair of subtasks processes each index; one subtask sorts extracted keys, while the other subtask builds the index. REBUILD INDEX begins building each index as soon as the corresponding sort generates its first sorted record. If you specify STATISTICS, a third subtask collects the sorted keys and updates the catalog table in parallel.

The subtasks that are used for the parallel REBUILD INDEX processing use Db2 connections. If you receive message DSNU397I that indicates that the REBUILD INDEX utility is constrained, increase the number of concurrent connections by using the MAX BATCH CONNECT parameter on panel DSNTIPE.

The greatest elapsed processing-time improvements result from parallel rebuilding for:
- Multiple indexes on a table space
- A partitioning index or a data-partitioned secondary index on all partitions of a partitioned table space
- A nonpartitioned secondary index on a partitioned table space

The following figure shows the flow of a REBUILD INDEX job with a parallel index build. The same flow applies whether you rebuild a data-partitioned secondary index or a partitioning index. Db2 starts multiple subtasks to unload the entire partitioned table space. Subtasks then sort index keys and build the partitioning index in parallel. If you specify STATISTICS, additional subtasks collect the sorted keys and update the catalog table in parallel, eliminating the
need for a second scan of the index by a separate RUNSTATS job.

The following figure shows the flow of a REBUILD INDEX job with a parallel index build. Db2 starts multiple subtasks to unload all partitions of a partitioned table space and to sort index keys in parallel. The keys are then merged and passed to the build subtask, which builds the nonpartitioned secondary index. If you specify STATISTICS, a separate subtask collects the sorted keys and updates the catalog table.

When parallel index build is used:
REBUILD INDEX always sorts the index keys and builds them in parallel for partitioned table spaces unless constrained by available memory, sort work files, or UTPRInnn file allocations.

Sort work data sets for parallel index build:
You can either allow the utility to dynamically allocate the data sets that SORT needs, or provide the necessary data sets yourself. Select one of the following methods to allocate sort work data sets and message data sets:
Method 1:
REBUILD INDEX determines the optimal number of sort work data sets and message data sets.

1. Specify the SORTDEVT keyword in the utility statement.
2. Allow dynamic allocation of sort work data sets by not supplying SORTWKnn DD statements in the REBUILD INDEX utility JCL.
3. Allocate UTPRINT to SYSOUT.

Method 2:
You control allocation of sort work data sets, and REBUILD INDEX allocates message data sets.

1. Provide DD statements with DD names in the form SWnnWKmm.
2. Allocate UTPRINT to SYSOUT.

Method 3:
You have the most control over rebuild processing; you must specify both sort work data sets and message data sets.

1. Provide DD statements with DD names in the form SWnnWKmm.
2. Provide DD statements with DD names in the form UTPRINnn.

Data sets that are used

If you select Method 2 or 3, define the necessary data sets by using the following information.

Each sort subtask must have its own group of sort work data sets and its own print message data set. In addition, you need to allocate the merge message data set when you build a single nonpartitioned secondary index on a partitioned table space.

Possible reasons to allocate data sets in the utility job JCL rather than using dynamic allocation are to:
• Control the size and placement of the data sets
• Minimize device contention
• Optimally use free disk space
• Limit the number of utility subtasks that are used to build indexes

The DD names SWnnWKmm define the sort work data sets that are used during utility processing. nn identifies the subtask pair, and mm identifies one or more data sets that are to be used by that subtask pair. For example:

SW01WK01
Is the first sort work data set that is used by the subtask that builds the first index.

SW01WK02
Is the second sort work data set that is used by the subtask that builds the first index.

SW02WK01
Is the first sort work data set that is used by the subtask that builds the second index.
SW02WK02
Is the second sort work data set that is used by the subtask that builds the second index.

The DD names UTPRINnn define the sort work message data sets that are used by the utility subtask pairs. nn identifies the subtask pair.

Every time you invoke REBUILD INDEX, new UTPRINnn data sets are dynamically allocated. REBUILD INDEX does not reuse UTPRINnn data sets from previous job steps. This behavior might cause the available JES2 job queue elements to be consumed more quickly than expected.

If you allocate the UTPRINT DD statement to SYSOUT in the job statement, the sort message data sets and the merge message data set, if required, are dynamically allocated. If you want the sort message data sets, merge message data sets, or both, allocated to a disk or tape data set rather than to SYSOUT, you must supply the UTPRINnn or the UTMERG01 DD statements (or both) in the utility JCL. If you do not allocate the UTPRINT DD statement to SYSOUT, and you do not supply a UTMERG01 DD statement in the job statement, partitions are not unloaded in parallel.

Determining the number of sort subtasks

The maximum number of utility subtasks that are started for parallel index build equals:

- For a simple table space, segmented table space, or simple partition of a partitioned table space, the number of indexes that are to be built
- For a single index that is being built on a partitioned table space, the number of partitions that are to be unloaded

REBUILD INDEX determines the number of subtasks according to the following guidelines:

- The number of subtasks equals the number of allocated sort work data set groups.
- The number of subtasks equals the number of allocated message data sets.
- If you allocate both sort work data sets and message data set groups, the number of subtasks equals the smallest number of allocated data sets.

Allocation of sort subtasks

REBUILD INDEX attempts to assign one sort subtask for each index that is to be built. If REBUILD INDEX cannot start enough subtasks to build one index per subtask, it allocates any excess indexes across the pairs (in the order that the indexes were created), so that one or more subtasks might build more than one index.

Estimating the sort work file size

If you choose to provide the data sets, you need to know the size and number of keys that are present in all of the indexes or index partitions that are being processed by the subtask in order to calculate each sort work file size. When you determine which indexes or index partitions are assigned to which subtask pairs, use the following formula to calculate the required space.

\[ 2 \times (\text{maximum record length} \times \text{numcols} \times (\text{count} + 2) \times \text{number of indexes}) \]
The variables in the preceding formula have the following values:

**maximum record length**
Maximum record length of the SYSCOLDISTSTATS record that is processed when collecting frequency statistics (You can obtain this value from the RECLENGTH column in SYSTABLES.)

**numcols**
Number of key columns to concatenate when you collect frequent values from the specified index.

**count**
Number of frequent values that Db2 is to collect.

### Overriding dynamic sort work data set allocation

Db2 estimates how many records are to be sorted. This information is used for dynamic allocation of sort work space. Sort work space is allocated by Db2 or by the sort program that is used.

If the table space contains rows with VARCHAR columns, Db2 might not be able to accurately estimate the number of records. If the estimated number of records is too high, if the requested sort work space is not available, or if the estimated number of records is too low, which causes the sort to overflow, the utility might fail and cause an abend.

**Recommendation:** To enable Db2 to calculate a more accurate estimate:

- For a table space that is partitioned (non-universal), run RUNSTATS UPDATE ALL before REORG.
- For any other type of table space, run RUNSTATS UPDATE SPACE before REORG.

When you run RUNSTATS with SHRLEVEL REFERENCE, real-time statistics values are also updated.

You can override the dynamic allocation of sort work space in one of the following ways:

- Allocate the sort work data sets with SORTWKnn DD statements in your JCL.
- If the number of keys for the affected index in column TOTALENTRIES of table SYSIBM.SYSINDEXSPACESTATS is not available or is significantly incorrect, you can update the value to a more appropriate value using an SQL UPDATE statement. When REBUILD INDEX on the affected index completes, TOTALENTRIES is set to the number of keys for the affected index.
- If the number of rows in the associated table space in column TOTALROWS of table SYSIBM.SYSTABLESPACESTATS is not available or is significantly incorrect, you can update the value to a more appropriate value using an SQL UPDATE statement. The next time that REORG is run, TOTALROWS is set to the number of rows in the associated table space.

**Related reference:**
- Db2 Sort for z/OS

**Related information:**
- DFSORT Application Programming Guide
Resetting the REBUILD-pending status

REBUILD-pending status (which appears as RBDP in the output from the DISPLAY command) means that the physical or logical index partition, nonpartitioned secondary index, or logical partition of a nonpartitioned secondary index is in REBUILD-pending status.

The variations of REBUILD-pending status are as follows:

**RBDP**
The physical or logical index partition is in the REBUILD-pending status. The individual physical or logical index partition is inaccessible. Reset the RBDP status by rebuilding the single affected partition. If multiple partitions are in RBDP status, you can rebuild either the entire index or all affected partitions.

**RBDP**
The logical partition of the nonpartitioned secondary index is in the REBUILD-pending status. The entire nonpartitioned secondary index is inaccessible. Reset RBDP status by rebuilding only the affected logical partitions.

**PSRBD**
The nonpartitioned secondary index space is in the REBUILD-pending status. The entire index space is inaccessible. Rebuild the object with the REBUILD INDEX utility. This state only applies to nonpartitioned secondary indexes.

You can reset the REBUILD-pending status for an index with any of these operations:
- REBUILD INDEX
- REORG TABLESPACE SORTDATA
- REPAIR SET INDEX with NORBDPEND
- START DATABASE command with ACCESS FORCE

**Attention:** Use the START DATABASE command with ACCESS FORCE only as a means of last resort.

Rebuilding critical catalog indexes

For information about how to rebuild critical catalog indexes, see the information about recovering catalog and directory objects.

Related tasks:
[“Recovering catalog and directory objects” on page 485](#)

Recoverability of a rebuilt index

When you successfully rebuild an index that was defined with COPY YES, utility processing inserts a SYSCOPY row with ICTYPE='B' for each rebuilt index. Rebuilt indexes are also placed in informational COPY-pending status, which indicates that you should make a copy of the index.

**Recommendation:** If you have FlashCopy capability, create a FlashCopy image copy during the REBUILD INDEX. Alternatively, after the index is rebuilt, make a sequential full image copy of the index to create a recovery point. Both copy methods reset the ICOPY status of the rebuilt index.
Creating a FlashCopy image copy with REBUILD INDEX

As part of REBUILD INDEX processing, you can use FlashCopy technology to quickly take image copies of the target objects.

About this task

Restriction: You cannot create FlashCopy image copies of indexes that are defined with the COPY NO attribute.

Procedure

To create a FlashCopy image copy with REBUILD INDEX:

Specify FLASHCOPY(YES) or FLASHCOPY(CONSISTENT) in the REBUILD INDEX utility control statement. Alternatively, you can set the FLASHCOPY_REBUILD_INDEX subsystem parameter to YES, which specifies that REBUILD INDEX is to use FLASHCOPY(YES) by default. The value that you specify for the FLASHCOPY option in the REBUILD INDEX statement always overrides the value for the FLASHCOPY_REBUILD_INDEX subsystem parameter. Optionally, you can also specify FCCOPYDDN in the REBUILD INDEX statement. Use this option to specify a template for the FlashCopy image copy. If you do not specify the FCCOPYDDN option in the REBUILD INDEX statement, the utility uses the value from the FCCOPYDDN subsystem parameter.

Restriction: The data sets that you specify for the FlashCopy image copy must be on FlashCopy Version 2 disk volumes.

When you specify FLASHCOPY(YES) or FLASHCOPY(CONSISTENT), REBUILD INDEX uses FlashCopy technology to create a consistent copy of the target objects. The FlashCopy image copy fails if the FlashCopy Version 2 disk volumes are not available or if any of the other FlashCopy operational restrictions exist. For a list of those operational restrictions, see “FlashCopy image copies” on page 157.

Related concepts:

“FlashCopy image copies” on page 157

Related reference:

 DEFAULT TEMPLATE field (FCCOPYDDN subsystem parameter) (Db2 Installation and Migration)

 REBUILD INDEX field (FLASHCOPY_REBUILD_INDEX subsystem parameter) (Db2 Installation and Migration)

Termination or restart of REBUILD INDEX

You can terminate and restart the REBUILD INDEX utility.

You can terminate REBUILD INDEX by using the TERM UTILITY command. If you terminate a REBUILD INDEX job, the index space is placed in the REBUILD-pending status and is unavailable until it is successfully rebuilt.

By default, Db2 uses RESTART(PHASE) when restarting REBUILD INDEX jobs. The job starts again from the beginning.

If you restart a job that uses the STATISTICS keyword, inline statistics collection does not occur. To update catalog statistics, run the RUNSTATS utility after the restarted REBUILD INDEX job completes.
The effect of REBUILD INDEX on index version numbers

Db2 stores the range of used index version numbers in the OLDEST_VERSION and CURRENT_VERSION columns of the SYSIBM.SYSINDEXES and SYSIBM.SYSINDEXEXPART catalog tables.

The OLDEST_VERSION column contains the oldest used version number, and the CURRENT_VERSION column contains the current version number.

When you run REBUILD INDEX, the utility updates this range of used version numbers for indexes that are defined with the COPY NO attribute. REBUILD INDEX sets the OLDEST_VERSION column to the current version number, which indicates that only one version is active; Db2 can then reuse all of the other version numbers.

Recycling of version numbers is required when all of the version numbers are being used. All version numbers are being used when one of the following situations is true:

- The value in the CURRENT_VERSION column is one less than the value in the OLDEST_VERSION column
- The value in the CURRENT_VERSION column is 15, and the value in the OLDEST_VERSION column is 0 or 1.

You can also run LOAD REPLACE, REORG INDEX, or REORG TABLESPACE to recycle version numbers for indexes that are defined with the COPY NO attribute. To recycle version numbers for indexes that are defined with the COPY YES attribute or for table spaces, run MODIFY RECOVERY.

Related concepts:

- Table space versions (Db2 Administration Guide)

Sample REBUILD INDEX control statements

Use the sample control statements as models for developing your own REBUILD INDEX control statements.

Example 1: Rebuilding an index

The following control statement specifies that the REBUILD INDEX utility is to rebuild the DSN8B10.XDEPT1 index.

```sql
//STEP1 EXEC DSNUPROC,UID='IUIQU2UT.RBLD1',TIME=1440,
// UTPROC=''
// SYSTEM='DSN'
//SYSREC DD DSN=IUIQU2UT.RBLD1.STEP1.SYSREC,DISP=(MOD,DELETE,CATLG),
// UNIT=SYSDA,SPACE=(8000,(20,20),,,ROUND)
//SYSIN DD *
REBUILD INDEX (DSN8B10.XDEPT1)
/*
Example 2: Rebuilding index partitions

The following control statement specifies that REBUILD INDEX is to rebuild partitions 2 and 3 of the DSN8B10.XEMP1 index. The partition numbers are indicated by the PART option.

REBUILD INDEX (DSN8B10.XEMP1 PART 2, DSN8B10.XEMP1 PART 3)

Example 3: Rebuilding multiple partitions of a partitioning or secondary index

The following control statement specifies that REBUILD INDEX is to rebuild partitions 2 and 3 of the DSN8B10.XEMP1 index. The partition numbers are indicated by the PART option. The SORTDEV'T and SORTNUM keywords indicate that the utility is to use dynamic data set and message set allocation. Parallelism is used by default.

If sufficient virtual storage resources are available, Db2 starts one pair of utility sort subtasks for each partition. This example does not require UTPRINT'nn DD statements because it uses DSNUPROC to invoke utility processing. DSNUPROC includes a DD statement that allocates UTPRINT to SYSOUT.

//SAMPJOB JOB ...
//STEP1 EXEC DSNUPROC,UID='SAMPJOB.RBINDEX',UTPROC='',SYSTEM='DSN'
//SYSIN DD *
REBUILD INDEX (DSN8B10.XEMP1 PART 2, DSN8B10.XEMP1 PART 3)
SORTDEV'T SYSWK
SORTNUM 4
/*

Example 4: Rebuilding all partitions of a partitioning index

The control statement specifies that REBUILD INDEX is to rebuild all index partitions of the DSN8B10.XEMP1 partitioning index. Parallelism is used by default. For this example, REBUILD INDEX allocates sort work data sets in two groups, which limits the number of utility subtask pairs to two. This example does not require UTPRINT'nn DD statements because it uses DSNUPROC to invoke utility processing. DSNUPROC includes a DD statement that allocates UTPRINT to SYSOUT.

//SAMPJOB JOB ...
//STEP1 EXEC DSNUPROC,UID='SAMPJOB.RCVINDEX',UTPROC='',SYSTEM='DSN'
/* First group of sort work data sets for parallel index rebuild
//SW01WK01 DD UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//SW01WK02 DD UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//SW01WK03 DD UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
/* Second group of sort work data sets for parallel index rebuild
//SW02WK01 DD UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//SW02WK02 DD UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//SW02WK03 DD UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//SYSIN DD *
REBUILD INDEX (DSN8B10.XEMP1)
/*

Figure 55. Example REBUILD INDEX statement
Example 5: Rebuilding all indexes of a table space

The following control statement specifies that REBUILD INDEX is to rebuild all indexes for table space DSN8D11A.DSN8S11E. The SORTDEVT and SORTNUM keywords indicate that the utility is to use dynamic data set and message set allocation. Parallelism is used by default.

If sufficient virtual storage resources are available, Db2 starts one utility sort subtask to build the partitioning index and another utility sort subtask to build the nonpartitioning index. This example does not require UTPRINnn DD statements because it uses DSNUPROC to invoke utility processing. DSNUPROC includes a DD statement that allocates UTPRINT to SYSOUT.

```
//SAMPJOB JOB ...
//STEP1 EXEC DSNUPROC,UID='SAMPJOB.RCVINDEX',UTPROC='',SYSTEM='DSN'
//SYSIN DD *
REBUILD INDEX (ALL) TABLESPACE DSN8D11A.DSN8S11E
SORTDEVT SYSWK
SORTNUM 4
/*
```

Example 6: Rebuilding indexes only if they are in a restrictive state and gathering inline statistics

The control statement in this example specifies that REBUILD INDEX is to rebuild partition 9 of index IDOS482D if it is in REBUILD-pending (RBDP), RECOVER-pending (RECP), or advisory REORG-pending (AREO*) state. This condition that the index be in a certain restrictive state is indicated by the SCOPE PENDING option. The STATISTICS FORCEROLLUP YES option indicates that the utility is to collect inline statistics on the index partition that it is rebuilding and to force aggregation of those statistics.

```
//STEP6 EXEC DSNUPROC,UID='JUOSU248.CHK6',
// UTPROC='',
// SYSTEM='SSTR'
//UTPRINT DD SYOUT=*
//SYSREC DD DSN=JUOSU248.CHKIXPX.STEP6.SYSREC,
// DISP=(MOD,DELETE,CATLG),
// UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSCOPY DD DSN=JUOSU248.CHKIXPX.STEP6.SYSCOPY,
// DISP=(MOD,DELETE,CATLG),
// UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SORTOUT DD DSN=JUOSU248.CHKIXPX.STEP6.SORTOUT,
// DISP=(MOD,DELETE,CATLG),
// UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSIN DD *
   REBUILD INDEX (IDOS482D PART 9)
   STATISTICS FORCEROLLUP YES
   SCOPE PENDING
*/
```

Figure 56. Example REBUILD INDEX statement with STATISTICS option

Example 7: Rebuilding indexes with SHRLEVEL CHANGE.

The following control statement specifies that during the rebuild, applications can read from and write to ADMF001.IUKQAI01.

```
REBUILD INDEX (ADMF001.IUKQAI01)
   SHRLEVEL CHANGE
```
Example 8: Rebuilding indexes that are on clone tables

The following control statement specifies that REBUILD INDEX is to reconstruct only the specified indexes that are on clone tables.

REBUILD INDEX (ADMF001.IUKQAI01)
CLONE
Chapter 26. RECOVER

The RECOVER utility recovers data to the current state or to a previous point in time by restoring a copy and then applying log records. The RECOVER utility can also recover data to a previous point in time by backing out committed work.

The largest unit of data recovery is the table space or index; the smallest is the page. You can recover a single object or a list of objects. The RECOVER utility recovers an entire table space, index, a partition or data set, pages within an error range, or a single page. You can recover data from sequential image copies of an object, a FlashCopy image copy of an object, a system-level backup, or the log. Point-in-time recovery with consistency automatically detects the uncommitted transactions that are running at the recover point in time and rolls back their changes on the recovered objects. After recovery, objects will be left in their transactionally consistent state.

You can use the RECOVER utility in conjunction with z/OS DFSMS data set encryption to encrypt or decrypt table spaces or indexes.

Output

Output from RECOVER consists of recovered data (a table space, index, partition or data set, error range, or page within a table space).

Authorization required

To run this utility, you must use a privilege set that includes one of the following authorities:

- RECOVERDB privilege for the database
- DBADM or DBCTRL authority for the database. If the object on which the utility operates is in an implicitly created database, DBADM authority on the implicitly created database or DSNDB04 is required.
- System DBADM authority
- DATAACCESS authority
- SYSCTRL or SYSADM authority

An ID with installation SYSOPR authority can also run RECOVER, but only on a table space in the DSNDB01 or DSNDB06 database.

Restrictions on running RECOVER

The following restrictions apply to the general use of the RECOVER utility.

- RECOVER cannot recover a table space or index that is defined to use a storage group that is defined with mixed specific and nonspecific volume IDs. If you specify such a table space or index, the job terminates and you receive error message DSNU419I.
- RECOVER cannot recover an index that was altered to PADDED or NOT PADDED. Instead, you need to rebuild the index.
Partition level recovery is required when the recovery base is an inline copy that was created by a REORG that pruned partitions.

### Execution phases of RECOVER

The RECOVER utility operates in these phases:

<table>
<thead>
<tr>
<th>Phase</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTILINIT</td>
<td>Performs initialization and setup.</td>
</tr>
<tr>
<td>RESTORE</td>
<td>Locates and merges any appropriate sequential image copies and restores the table space to a backup level; processes a list of objects in parallel if you specify the PARALLEL keyword.</td>
</tr>
<tr>
<td>RESTORER</td>
<td>If you specify the PARALLEL keyword, reads and merges the sequential image copies.</td>
</tr>
<tr>
<td>RESTOREW</td>
<td>If you specify the PARALLEL keyword, writes the pages to the object.</td>
</tr>
<tr>
<td>PRELOGC</td>
<td>Preliminary LOGCSR phase. Determines uncommitted work that was backed out when the recovery base for an object is a FlashCopy image copy with consistency.</td>
</tr>
<tr>
<td>PRELOGA</td>
<td>Preliminary LOGAPPLY phase. Applies the uncommitted work up to the point of consistency for the object with a FlashCopy image copy with consistency recovery base.</td>
</tr>
<tr>
<td>LOGAPPLY</td>
<td>Applies any outstanding log changes to the object that is restored from the previous phase or step. If a recover job fails in the middle of the LOGAPPLY phase, it can be restarted from last commit point.</td>
</tr>
<tr>
<td>LOGCSR</td>
<td>Analyzes log records and constructs information about inflight, indoubt, inabort, and postponed abort units of recovery. This phase is executed if either the TORBA and TOLOGPOINT option was specified. If a recover job fails in the middle of the LOGCSR phase, it can be restarted from the beginning of the LOGCSR phase. Db2 members that finished the LOGCSR phase before the RECOVER job failure go through the LOGCSR phase again. For BACKOUT YES processing, LOGCSR analyzes log records and constructs information about committed and canceled units of recovery.</td>
</tr>
<tr>
<td>LOGUNDO</td>
<td>Rolls back any uncommitted changes that the active units of recovery made to the recovered objects. This phase is executed if either the TORBA and TOLOGPOINT option was specified. If you need to restart the recover job after it enters into the LOGUNDO phase, objects that were not changed by URs that were active during the recover to point in time will be marked as finished and no need for further processing.</td>
</tr>
</tbody>
</table>
For BACKOUT YES processing, the LOGUNDO phase backs out committed changes from the current state of the object to the prior point in time specified. In addition, any uncommitted changes at the point in time specified are rolled back.

**UTILTERM**
Performs cleanup.

---

**Syntax and options of the RECOVER control statement**

The RECOVER utility control statement, with its multiple options, defines the function that the utility job performs.

You can create a control statement with the ISPF/PDF edit function. After creating it, save it in a sequential or partitioned data set. When you create the JCL for running the job, use the SYSIN DD statement to specify the name of the data set that contains the utility control statement.

**Syntax diagram**

![Syntax diagram](image)

**Notes:**

1. DSNUM integer is not valid for nonpartitioned indexes.
2. Use the LOGRANGES NO option only at the direction of IBM Support. This option can cause the LOGAPPLY phase to run much longer and, in some cases, apply log records that should not be applied.

**object:**

![Object diagram](image)
Option descriptions

You can specify a list of objects by repeating the TABLESPACE, INDEX, or INDEXSPACE keywords. If you use a list of objects, the valid keywords are: DSNUM, TORBA, TOLOGPOINT, LOGONLY, PARALLEL, and either LOCALSITE or RECOVERYSITE.

The options TOCOPY, TOLASTCOPY, TOLASTFULLCOPY, TORBA and TOLOGPOINT are all referred to as point-in-time recovery options.

**LIST listdef-name**

Specifies the name of a previously defined LISTDEF list name. The utility allows one LIST keyword for each control statement of RECOVER. The list can contain a mixture of table spaces and index spaces. RECOVER is invoked once for the entire list.

This utility does not support lists that specify more than 32,000 objects.

Partitions of table spaces or index spaces that are included by the PARTLEVEL keyword count as separate objects.

This utility only processes clone data if the CLONE keyword is specified. The use of CLONED YES on the LISTDEF statement is not sufficient.

The partitions or partition ranges can be specified in a list.
TABLESPACE  

`database-name.table-space-name`

Specifies the table space (and optionally, the database to which it belongs) that is to be recovered.

You can specify a list of table spaces by repeating the TABLESPACE keyword. You can recover an individual catalog or directory table space in a list with its IBM-defined indexes.

`database-name`

Is the name of the database to which the table space belongs.

The default value is **DSNDB04**.

`table-space-name`

Is the name of the table space that is to be recovered.

INDEXSPACE  

`database-name.index-space-name`

Specifies the index space that is to be recovered.

`database-name`

Specifies the name of the database to which the index space belongs.

The default value is **DSNDB04**.

`index-space-name`

Specifies the name of the index space that is to be recovered.

INDEX  

`creator-id.index-name`

Specifies the index in the index space that is to be recovered. The RECOVER utility can recover only indexes that were defined with the COPY YES attribute and subsequently copied.

`creator-id`

Optionally specifies the creator of the index.

The default value is the user identifier for the utility.

`index-name`

Specifies the name of the index in the index space that is to be recovered. Enclose the index name in quotation marks if the name contains a blank.

DSNUM

Identifies a partition within a partitioned table space or a partitioned index, or identifies a data set within a nonpartitioned table space that is to be recovered. Alternatively, the option can recover the entire table space or index space.

You cannot specify a logical partition of a nonpartitioned index. You cannot specify a single data set of a nonpartitioned index unless the data set that is specified for TOCOPY is a FlashCopy image copy. However, to ensure consistency, all data sets of a nonpartitioned index should be recovered to the same point in time.

**ALL**

Specifies that the entire table space or index space is to be recovered.

`integer`

Specifies the number of the partition or data set that is to be recovered.

The maximum value is 4096.

For a partitioned table space or index space:

The integer is its physical partition number.

For a nonpartitioned table space:
Find the integer at the end of the data set name. The data set name has the following format:

```
catname.DSNDbx.dbname.tsname.y000z.Annn
```

- `catname` is the VSAM catalog name or alias.
- `x` is C or D.
- `dbname` is the database name.
- `tsname` is the table space name.
- `y` is I or J.
- `z` is 1 or 2.
- `nnn` is the data set integer.

**PAGE** `page-number`

Specifies a particular page that is to be recovered. You cannot specify this option if you are recovering from a concurrent copy.

`page-number` is the number of the page, in either decimal or hexadecimal notation. For example, both 999 and X'3E7' represent the same page. PAGE is invalid with the LIST specification.

**CONTINUE**

Specifies that the recovery process is to continue. Use this option only if an error causes RECOVER to terminate during reconstruction of a page. In this case, the page is marked as “broken”. After you repair the page, you can use the CONTINUE option to recover the page, starting from the point of failure in the recovery log.

Contact IBM Support before you run the RECOVER utility with the PAGE CONTINUE keywords.

**TORBA** `X'byte-string'`

Specifies, in a non-data-sharing environment, a point on the log to which RECOVER is to recover. Specify an RBA value. The recovery process ends with the last log record whose relative byte address (RBA) is not greater than `X'byte-string'`. If `X'byte-string'` is the RBA of the first byte of a log record, that record is included in the recovery.

The RBA is a string of up to 20 hexadecimal characters, which represent the 6-byte RBA format or the extended 10-byte RBA format. Values are padded on the left with zeros if needed. Any 6-byte values are immediately converted to 10-byte format. All further RECOVER processing uses the 10-byte format.

In a data sharing environment, use TORBA only when you want to recover to a point before the originating member joined the data sharing group. If you specify an RBA after this point, the recovery fails.

For a NOT LOGGED table space, the value must be a recoverable point.

Uncommitted work by units of recovery that are active at the specified RBA are backed out by RECOVER so that each object is left in a consistent state.

**TOLOGPOINT** `X'byte-string'`

Specifies a point on the log to which RECOVER is to recover. Specify either an RBA or an LRSN value.

The RBA or LRSN is a string of 12 or 20 hexadecimal characters, which represent the 6-byte RBA format or the extended 10-byte RBA format. In a
data-sharing environment, a 6-byte LRSN value must be greater than 'X'00FFFFFF00' and a 10-byte value must be greater than 'X'0000FFFFFFFFF00'.

For a NOT LOGGED table space, the value must be a recoverable point.

Uncommitted work by units of recovery that are active at the specified LRSN or RBA will be backed out by RECOVER, leaving each object in a consistent state.

**REUSE**
Specifies that RECOVER is to logically reset and reuse Db2-managed data sets without deleting and redefining them. If you do not specify REUSE, Db2 deletes and redefines Db2-managed data sets to reset them.

If you are recovering an object because of a media failure, do not specify REUSE.

If a data set has multiple extents, the extents are not released if you use the REUSE parameter.

**CURRENTCOPYONLY**
Specifies that RECOVER is to improve the performance of restoring concurrent copies (copies that were made by the COPY utility with the CONCURRENT option) by using only the most recent primary copy for each object in the list.

When you specify CURRENTCOPYONLY for a concurrent copy, RECOVER builds a DFSMSdss RESTORE command for each group of objects that is associated with a concurrent copy data set name. If the RESTORE fails, RECOVER does not automatically use the next most recent copy or the backup copy, and the object fails. If you specify DSNUM ALL with CURRENTCOPYONLY and one partition fails during the restore process, the entire utility job on that object fails.

If you specify CURRENTCOPYONLY and the most recent primary copy of the object to be recovered is not a concurrent copy, Db2 ignores this keyword.

For objects in the recovery list whose recovery base is a system-backup, the default is CURRENTCOPYONLY.

**PARALLEL**
Specifies the maximum number of objects in the list that are to be restored in parallel from image copies on disk or tape. RECOVER attempts to retain tape mounts for tapes that contain stacked image copies when the PARALLEL keyword is specified. In addition, to maximize performance, RECOVER determines the order in which objects are to be restored. PARALLEL also specifies the maximum number of objects in the list that are to be restored in parallel from system-level backups that have been dumped to tape. The processing may be limited by DFSMShsm.

If you specify TAPEUNITS with PARALLEL, you control the number of tape drives that are dynamically allocated for the recovery function. The TAPEUNITS keyword applies only to tape drives that are dynamically allocated. The TAPEUNITS keyword does not apply to JCL-allocated tape drives. The total number of tape drives that are allocated for the RECOVER job is the sum of the JCL-allocated tape drives, and the number of dynamically allocated tape drives. The number of dynamically allocated tape drives is determined as follows:

- The specified value for TAPEUNITS.
The value that is determined by the RECOVER utility if you omit the TAPEUNITS keyword. The number of tape drives that RECOVER attempts to allocate is determined by the object in the list that requires the most tape drives.

If you specify PARALLEL, you cannot specify TOCOPY, TOLASTCOPY, or TOLASTFULLCOPY.

\((\text{num-objects})\)

Specifies the number of objects in the list that are to be processed in parallel. If storage constraints are encountered, you can adjust this value to a smaller value.

If you specify 0 or do not specify TAPEUNITS keyword, RECOVER determines the optimal number of objects to process in parallel.

**TAPEUNITS**

Specifies the number of tape drives that the utility should dynamically allocate for the list of objects that are to be processed in parallel. If you omit this keyword, the utility determines the number of tape drives to allocate for the recovery function.

The TAPEUNITS option does not apply to recovery from system-level backups. In this case, DFSMShsm determines the number of tape drives that are used for the recovery.

\((\text{num-tape-units})\)

Specifies the number of tape drives to allocate. If you specify 0 or do not specify a value for num-tape-units, RECOVER determines the maximum number of tape units to use at one time. RECOVER TAPEUNITS has a max value of 32767.

**RESTOREBEFORE X’byte-string’**

Specifies that RECOVER is to search for an image copy, concurrent copy, or system-level backup (if yes has been specified for the SYSTEM_LEVEL_BACKUPS subsystem parameter) with an RBA or LRSN value earlier than the specified X’byte-string’ value to use in the RESTORE phase.

The RBA or LRSN is a string of 12 or 20 hexadecimal characters, which represent the 6-byte RBA format or the extended 10-byte RBA format. Any 10-byte values are immediately converted to 6-byte format. All further RECOVER processing is performed with the 6-byte format. In a data-sharing environment, a 6-byte LRSN value must be greater than X’00FFFFFFFFFFFFFF’ and a 10-byte value must be greater than X’0000FFFFFFFFFFFF’.

To avoid specific image copies, concurrent copies, or system-level backups with matching or more recent RBA or LRSN values in START_RBA, the RECOVER utility applies the log records and restores the object to its current state or the specified TORBA or TOLOGPOINT value. The RESTOREBEFORE value is compared with the RBA or LRSN value in the START_RBA column in the SYSIBM.SYSCOPY record for those copies. For system-level backups, the RESTOREBEFORE value is compared with the data complete LRSN.

If you specify a TORBA or TOLOGPOINT value with the RESTOREBEFORE option, the RBA or LRSN value for RESTOREBEFORE must be lower than the specified TORBA OR TOLOGPOINT value. If you specify RESTOREBEFORE, you cannot specify TOCOPY, TOLASTCOPY, or TOLASTFULLCOPY.
FROMDUMP
Specifies that only dumps of the database copy pool are used for the restore of the data sets.

DUMPCCLASS (dcl)
Indicates the DFSMShsms dump class to use to restore the data sets.

The FROMDUMP and DUMPCCLASS options that you specify for the RECOVER utility override the RESTORE/RECOVER and DUMPCCLASS NAME installation options that you specify on installation panel DSNTIP6.

LOGONLY
Specifies that the target objects are to be recovered from their existing data sets by applying only log records to the data sets. Db2 applies all log records that were written after a point that is recorded in the data set itself.

To recover an index space by using RECOVER LOGONLY, you must define the index space with the COPY YES attribute.

Use the LOGONLY option when the data sets of the target objects have already been restored to a point of consistency by another process offline, such as DFSMSdss concurrent copy.

LOGONLY is not allowed on a table space or index space with the NOT LOGGED attribute.

TOCOPY data-set
Specifies the particular image copy data set that Db2 is to use as a source for recovery.

data-set is the name of the data set.

If the data set is a full image copy, it is the only data set that is used in the recovery. If it is an incremental image copy, RECOVER also uses the previous full image copy and any intervening incremental image copies.

If you specify the data set as the local backup copy, Db2 first tries to allocate the local primary copy. If the local primary copy is unavailable, Db2 uses the local backup copy.

If you use TOCOPY or TORBA to recover a single data set of a nonpartitioned table space, Db2 issues message DSNU520I to warn that the table space can become inconsistent following the RECOVER job. This point-in-time recovery can cause compressed data to exist without a dictionary or can even overwrite the data set that contains the current dictionary.

If you use TOCOPY with a particular partition or data set (identified with DSNUM), the image copy must be for the same partition or data set, or for the whole table space or index space. If you use TOCOPY with DSNUM ALL, the image copy must be for DSNUM ALL. You cannot specify TOCOPY with a LIST specification. If the image copy is a Flash Copy image copy data set, and the object is partitioned, you must specify the number of the partition that is to be recovered on the DSNUM parameter.

If the image copy data set is a z/OS generation data set, supply a fully qualified data set name, including the absolute generation and version number.

If the NOSYSCOPY option is specified, you can specify a generation data set with a relative generation number.

If the image copy data set is not a generation data set and more than one image copy data set with the same data set name exists, use one of the following options to identify the data set exactly:
NOSYSCOPY
Specifies that the full image copy data set is not registered in the
SYSIBM.SYSCOPY catalog table. RECOVER will not verify the image copy.

You can specify a generation data set with a relative generation number. For
example, you can specify 'DSNC000.DBPAY.TSPAY.FULL(0)'.

Restriction: Concurrent image copies are not supported.

INLCOPY
The specified data set is an inline image copy that was created by a
REORG or LOAD REPLACE job; therefore, the data set might have
duplicated and out-of-order pages.

FCCOPY
The specified data set is a FlashCopy image copy that was created by a
COPY, REORG TABLESPACE, REORG INDEX, LOAD, or REBUILD
INDEX job with the FLASHCOPY YES or CONSISTENT options.

TOVOLUME
Identifies the image copy data set.

CATALOG
Indicates that the data set is cataloged. Use this option only for an image
copy that was created as a cataloged data set. (Its volume serial is not
recorded in SYSIBM.SYSCOPY.)

RECOVER refers to the SYSIBM.SYSCOPY catalog table during execution.
If you use TOVOLUME CATALOG, the data set must be cataloged. If you
remove the data set from the catalog after creating it, you must catalog the
data set again to make it consistent with the record for this copy that
appears in SYSIBM.SYSCOPY.

vol-ser
Identifies the data set by an alphanumeric volume serial identifier of its
first volume. Use this option only for an image copy that was created as a
noncataloged data set. Specify the first vol-ser in the SYSCOPY record to
locate a data set that is stored on multiple tape volumes.

TOSEQNO integer
Identifies the image copy data set by its file sequence number. integer is
the file sequence number.

TOLASTCOPY
Specifies that RECOVER is to restore the object to the last image copy that was
taken. If the last image copy is a full image copy, it is restored to the object. If
the last image copy is an incremental image copy, the most recent full copy
along with any incremental copies are restored to the object.

If the image copy is a Flash Copy image copy data set, and the object is
partitioned, you must specify the number of the partition that is to be
recovered on the DSNUM parameter.

TOLASTFULLCOPY
Specifies that the RECOVER utility is to restore the object to the last full image
copy that was taken. Any incremental image copies that were taken after the
full image copy are not restored to the object.

If the image copy is a Flash Copy image copy data set, and the object is
partitioned, you must specify the number of the partition that is to be
recovered on the DSNUM parameter.
ERROR RANGE

Specifies that all pages within the range of reported I/O errors are to be recovered. Recovering an error range is useful when the range is small, relative to the object that contains it; otherwise, recovering the entire object is preferred. You cannot specify this option if you are recovering from a concurrent copy.

In some situations, recovery using the ERROR RANGE option is not possible, such as when a sufficient quantity of alternate tracks cannot be obtained for all bad records within the error range. You can use the IBM Device Support Facility, ICKDSF service utility to determine whether this situation exists. In such a situation, redefine the error data set at a different location on the volume or on a different volume, and then run the RECOVER utility without the ERROR RANGE option.

You cannot specify ERROR RANGE with a LIST specification.

VERIFYSET

Specifies whether the RECOVER utility verifies that all related objects that are required for a point-in-time recovery are included in the RECOVER control statement. This option applies to point-in-time recoveries of base objects and the following related objects:

- LOB objects
- XML objects
- History objects
- Archive objects

The VERIFYSET option does not apply to point-in-time recoveries of catalog and directory objects. VERIFYSET NO behavior is always in effect for point-in-time recoveries of catalog and directory objects.

VERIFYSET YES

The RECOVER utility verifies that all related objects that are required to perform a point-in-time recovery are included in the RECOVER control statement. VERIFYSET YES is the default.

VERIFYSET NO

The RECOVER utility does not verify that all related objects that are required to perform a point-in-time recovery are included in the RECOVER control statement.

By specifying VERIFYSET NO, you can break up a point-in-time recovery into multiple jobs or avoid recovering objects that have changed since the selected recovery point.

Related information:

[Archive-enabled tables and archive tables (Introduction to Db2 for z/OS)]

ENFORCE YES

Specifies that CHKP and ACHKP pending states are set for a point-in-time recovery when only a subset of the related objects (BASE, LOB, XML, and RI) have been recovered to a point in time. ENFORCE YES is the default for catalog and directory objects. There is no override for the ENFORCE YES option for catalog and directory objects.

ENFORCE NO

Specifies that CHKP and ACHKP pending states are not set for a point-in-time recovery when only a subset of the related objects (BASE, LOB, XML, and RI) have been recovered to a point in time.
CLONE
Indicates that RECOVER is to recover only clone table data in the specified
table spaces, index spaces or indexes that contain indexes on clone tables. This
utility will only process clone data if the CLONE keyword is specified. The use
of CLONED YES on the LISTDEF statement is not sufficient.

LOCALSITE
Specifies that RECOVER is to use image copies from the local site. If you
specify neither LOCALSITE or RECOVERYSITE, RECOVER uses image copies
from the current site of invocation. (The current site is identified on the
installation panel DSNTIPO under SITE TYPE and in the macro DSN6SPRM
under SITETYP.)

RECOVERYSITE
Specifies that RECOVER is to use image copies from the recovery site. If you
specify neither LOCALSITE or RECOVERYSITE, RECOVER uses image copies
from the current site of invocation. (The current site is identified on the
installation panel DSNTIPO under SITE TYPE and in the macro DSN6SPRM
under SITETYP.)

LOGRANGES YES
Specifies that RECOVER should use SYSLGRNX information for the
LOGAPPLY phase. This option is the default.

LOGRANGES NO
Specifies that RECOVER should not use SYSLGRNX information for the
LOGAPPLY phase. Use this option only under the direction of IBM Support.

This option can cause RECOVER to run much longer. In a data sharing
environment this option can result in the merging of all logs from all members
that were created since the last image copy.

This option can also cause RECOVER to apply logs that should not be applied.
For example, assume that you take an image copy of a table space and then
run REORG LOG YES on the same table space. Assume also that the REORG
utility abends and you then issue the TERM UTILITY command for the
REORG job. The SYSLGRNX records that are associated with the REORG job
are deleted, so a RECOVER job with the LOGRANGES YES option (the
default) skips the log records from the REORG job. However, if you run
RECOVER LOGRANGES NO, the utility applies these log records.

BACKOUT
Specifies whether a log-only backout is to be used to recover objects to a prior
point in time. A log-only backout might decrease the amount of time that an
object is unavailable during a point-in-time recovery if the specified recovery
point is relatively recent.

NO Specifies that backout processing is not to be used.

BACKOUT NO is the default behavior.

YES
Specifies that RECOVER is to use the log to back out changes that were
made since the recovery point. (The recovery point is specified by the
TOLOGPOINT or TORMA options.) The changes are backed out from the
current state of the object. No image copy is restored. Any uncommitted
work at the specified recovery point is backed out so that the objects are
transactionally consistent.
If you specify BACKOUT YES, the recovery point must be within the most recent Db2 system checkpoints that are recorded in the BSDS for each member. Otherwise, the recovery cannot proceed and returns an error.

If you specify the BACKOUT keyword without YES or NO, YES is the default. (If you do not specify the BACKOUT keyword, BACKOUT NO is the default.)

Related information:
“Point-in-time recovery” on page 495

Before running RECOVER

Certain activities might be required before you run the RECOVER utility, depending on your situation.

If the table space or index space to be recovered is associated with a storage group, Db2 deletes and redefines the necessary data sets. If the STOGROUP has been altered to remove the volume on which the table space or index space is located, RECOVER places the data set on another volume of the storage group.

If you are using Flash Copy image copies, before you start the RECOVER utility confirm that the image copies are available in disk storage. If any of the required Flash Copy image copies have been migrated to tape, issue the DFSMShsm RECALL command to restore the image copies from tape to DASD.

Recovering data and indexes

You do not always need to recover both the data and indexes. If you recover the table space or index space to a current RBA or LRSN, any referentially related objects do not need to be recovered. If you plan to recover a damaged object to a point in time, use a consistent point in time for all of its referentially related objects, including related LOB and XML table spaces, for optimal performance. You must rebuild the indexes from the data if one of the following conditions is true:

- The table space is recovered to a point in time.
- An index is damaged.
- An index is in REBUILD-pending status.
- No image copy of the index is available.

If you need to recover both the data and the indexes, and no image copies of the indexes are available:
1. Use RECOVER TABLESPACE to recover the data.
2. Run REBUILD INDEX on any related indexes to rebuild them from the data.

If you have image copies of both the table spaces and the indexes, you can recover both sets of objects in the same RECOVER utility statement. The objects are recovered from the image copies and logs.

Data sets that RECOVER uses

The RECOVER utility uses a number of data sets during its operation.

The following table lists the data sets that RECOVER uses. The table lists the DD name that is used to identify the data set, a description of the data set, and an indication of whether it is required. Include statements in your JCL for each required data set and any optional data sets that you want to use.
Table 63. Data sets that RECOVER uses

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSIN</td>
<td>Input data set that contains the utility control statement.</td>
<td>Yes</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>Output data set for messages.</td>
<td>Yes</td>
</tr>
<tr>
<td>auth-id.job-name.HSM</td>
<td>A temporary data set that is automatically allocated by the utility and deleted when the utility completes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

The following objects are named in the utility control statement and do not require DD statements in the JCL:

**Table space, index space, or index**
Object that is to be recovered. If you want to recover less than an entire table space:
• Use the DSNUM option to recover a partition or data set.
• Use the PAGE option to recover a single page.
• Use the ERROR RANGE option to recover a range of pages with I/O errors.

**Image copy data set**
Copy that RECOVER is to restore. Db2 accesses this information through the Db2 catalog.

**System-level backups**
The RECOVER utility chooses the most recent backup (a sequential image copy, a concurrent copy, a FlashCopy image copy, or a system-level backup) to restore based on the recovery point for the table spaces or indexes (with the COPY YES attribute) being recovered.

Related concepts:
- "Before running RESTORE SYSTEM" on page 734
- "How the RECOVER utility retains tape mounts" on page 509
- "Data sets that online utilities use" on page 16

Related tasks:
- "Recovering with a system-level backup" on page 475

**Concurrency and compatibility for RECOVER**
The RECOVER utility has certain concurrency and compatibility characteristics associated with it.

Db2 treats individual data and index partitions as distinct target objects. Utilities that operate on different partitions of the same table space or index space are compatible. However, if a nonpartitioned secondary index exists on a partitioned table space, utilities that operate on different partitions of a table space can be incompatible because of contention on the nonpartitioned secondary index.

The following table shows which claim classes RECOVER claims and drains and any restrictive state that the utility sets on the target object.
Claims

Table 64. Claim classes of RECOVER operations.

<table>
<thead>
<tr>
<th>Target</th>
<th>RECOVER (no option)</th>
<th>RECOVER TORBA or TOCOPY</th>
<th>RECOVER PART TORBA or TOCOPY</th>
<th>RECOVER ERROR-RANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table space or partition</td>
<td>DA/UTUT</td>
<td>DA/UTUT</td>
<td>DA/UTUT</td>
<td>DA/UTUT CW/UTRW</td>
</tr>
<tr>
<td>Partitioning index, data-partitioned secondary index, or physical partition</td>
<td>DA/UTUT</td>
<td>DA/UTUT</td>
<td>DA/UTUT</td>
<td>DA/UTUT CW/UTRW</td>
</tr>
<tr>
<td>Nonpartitioned secondary index</td>
<td>DA/UTUT</td>
<td>DA/UTUT</td>
<td>DA/UTUT</td>
<td>DA/UTUT CW/UTRW</td>
</tr>
<tr>
<td>RI dependents</td>
<td>none</td>
<td>CHKP (YES)</td>
<td>CHKP (YES)</td>
<td>none</td>
</tr>
</tbody>
</table>

Legend:
- CHKP (YES): Concurrently running applications enter CHECK-pending after commit
- CW: Claim the write claim class
- DA: Drain all claim classes, no concurrent SQL access
- DR: Drain the repeatable read class, no concurrent access for SQL repeatable readers
- RI: Referential integrity
- UTRW: Utility restrictive state, read-write access allowed
- UTUT: Utility restrictive state, exclusive control
- none: Object is not affected by this utility

Note:
1. During the UTILINIT phase, the claim and restrictive states change from DA/UTUT to CW/UTRW.
2. Includes document ID indexes and node ID indexes over nonpartitioned XML table spaces and XML indexes.
3. Includes document ID indexes and node ID indexes over partitioned XML table spaces.

RECOVER does not set a utility restrictive state if the target object is DSNDB01.SYSUTILX.

Compatibility

The following table shows which utilities can run concurrently with RECOVER on the same target object. The target object can be a table space, an index space, or a partition of a table space or index space. If compatibility depends on particular options of a utility, that information is also documented in the table.

Table 65. Compatibility of RECOVER with other utilities

<table>
<thead>
<tr>
<th>Action</th>
<th>Compatible with RECOVER (no option)?</th>
<th>Compatible with RECOVER TORBA or TOCOPY?</th>
<th>Compatible with RECOVER ERROR-RANGE?</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHECK DATA</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CHECK INDEX</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CHECK LOB</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>COPY INDEXSPACE</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>COPY TABLESPACE</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DIAGNOSE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Action</td>
<td>Compatible with RECOVER (no option)?</td>
<td>Compatible with RECOVER TOCOPY or TORBA?</td>
<td>Compatible with RECOVER ERROR-RANGE?</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------</td>
<td>------------------------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>COPYTOCOPY</td>
<td>No</td>
<td>No</td>
<td>YNo</td>
</tr>
<tr>
<td>LOAD</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MERGECOPY</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MODIFY RECOVERY</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MODIFY STATISTICS</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>QUIESCE</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REBUILD INDEX</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REORG INDEX</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>REORG TABLESPACE</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REPAIR LOCATE INDEX</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>REPAIR LOCATE TABLESPACE</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REPORT</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RUNSTATS INDEX</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RUNSTATS TABLESPACE</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>STOSPACE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UNLOAD</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

To run on DSND801.SYSUTILX, RECOVER must be the only utility in the job step and the only utility running in the Db2 subsystem.

RECOVER on any catalog or directory table space is an exclusive job; such a job can interrupt another job between job steps, possibly causing the interrupted job to time out.

**Recovering with a system-level backup**

You can take system-level backups by using the BACKUP SYSTEM utility. In some cases, the RECOVER utility can use a system-level backup of the database copy pool as a recovery base.

**Before you begin**

Recovery of objects from system-level backups requires z/OS Version 1.8 or later.

**Procedure**

To specify that system-level backups can be used by RECOVER:

Specify YES for the SYSTEM_LEVEL_BACKUPS installation option on installation panel DSNTIP6.

The RECOVER utility chooses the most recent backup to restore. The backup can be:

- A sequential format image copy
- A VSAM FlashCopy® image copy
• A concurrent copy
• A system-level backup

The choice of which backup is to be restored is based on the recovery point for the table spaces or indexes that are being recovered. (For an index to be recovered in this way, the COPY YES attribute must be specified.) However, several exceptions for using a system-level backup exist.

Exceptions:
If any of the following utilities were run since the system-level backup that was chosen as the recovery base, the use of the system-level backup by the RECOVER utility is prohibited:
• REORG TABLESPACE
• REORG INDEX
• REBUILD INDEX
• LOAD REPLACE
• RECOVER from image copy or concurrent copy

In these cases, the recovery terminates with message DSNU1528I and return code 8.

Note: For z/OS Version 1 Release 11 and later, the RECOVER utility can use a system-level backup, even if the REBUILD INDEX, RECOVER, REORG, and LOAD utilities ran after the system-level backup was created. The RECOVER utility has been modified so that you can use system-level backups, even if a data set has moved since the backup was created.

If a REORG that removes dropped columns has run since the system-level backup that was chosen as the recovery base, the use of the system-level backup by the RECOVER utility is prohibited, and the recovery terminates with message DSNU556I and return code 8.

For a partition-by-growth table space, a point-in-time recovery is not allowed if the recovery period includes REORG TABLESPACE deleting empty partitions. REORG TABLESPACE can delete the highest numbered partitions if they are empty and the REORG_DROP_PBG_PRTS subsystem parameter is set to ENABLE.

The RECOVER utility invokes DFSMShsm to restore the data sets for the object from the system-level backup of the database copy pool. The user ID that invokes the RECOVER utility must have the authority to execute the DFSMShsm FRRECOV command.

Related reference:

FRRECOV command: Requesting a fast replication recovery (z/OS DFSMShsm Storage Administration Reference)

How to determine which system-level backups Db2 recovers

Db2 recovers different system level backups, depending on your situation.

To determine which system-level backups will be recovered:
• If you specify YES in the RESTORE/RECOVER FROM DUMP field on installation panel DSNTIP6 or you specify the FROMDUMP option in the RECOVER utility statement, Db2 uses only the dumps on tape of the database copy pool.
• If you specify a dump class name in the DUMP CLASS NAME field on installation panel DSNTIP6 or you specify the DUMPCLASS option in the RECOVER utility statement, Db2 uses dumps on tape of the database copy pool to restore the data sets from the DFSMShsm dump class.
- If you do not specify a dump class name in the DUMP CLASS NAME field on installation panel DSNTIP6, or you do not specify the DUMPCLASS option in the RECOVER utility statement, RESTORE SYSTEM issues the DFSMShsm LIST COPYPOOL command and uses the first dump class listed in the output.

If FROMDUMP was not specified on the RECOVER utility statement or on installation panel DSNTIP6, the system-level backup on disk is used. If the system-level backup does not reside on disk, an error message is issued. If FROMDUMP was specified either on the RECOVER utility statement or on installation panel DSNTIP6, then the dumped copy of the system-level backup on tape is used.

### Determining which recovery base Db2 uses

The recovery base is the copy that the RECOVER utility starts with when recovering an object. RECOVER then applies logs as needed.

**Procedure**

To determine which recovery base Db2 uses:

Run the REPORT utility with the RECOVERY option. Review the output to determine whether the objects to be recovered have any of the following items:

- Sequential image copies
- Concurrent copies
- FlashCopy image copies
- a utility LOG YES event

If you take system-level backups and the value of the SYSTEM_LEVEL_BACKUPS subsystem parameter is YES, also look at your system-level backup information in the BSDS to determine the recovery base.

**Related concepts:**

- “REPORT output” on page 710
- “How to determine which system-level backups Db2 recovers” on page 476

**Related reference:**

- SYSTEM-LEVEL BACKUPS field (SYSTEM_LEVEL_BACKUPS subsystem parameter) (Db2 Installation and Migration)

### Determining whether the system-level backups reside on disk or tape

Restoring data sets for objects in the database copy pool that are to be recovered from a system-level backup on disk occurs virtually instantaneously. Restoring data sets for objects that are to be recovered from a system-level backup on tape volumes takes much longer.

**Procedure**

To determine whether the system-level backups of the database copy pool reside on the disk or tape:

1. Run the DFSMShsm LIST COPYPOOL command with the ALLVOLS option.
2. Run the DSNJU004 utility output. For data sharing, run the DSNJU004 utility output on each member.
3. Review the output from the DFSMShsm LIST COPYPOOL command with the ALLVOLS option.

4. Review the Db2 system-level backup information in the DSNJU004 utility output.

**Results**

If the system-level backup that was chosen as the recovery base for the database copy pool no longer resides on DASD and the FROMDUMP option has not been specified, the recovery of the object will fail. You can specify the RECOVER FROMDUMP option, or specify it on installation panel DSNTIP6, to direct the utility to use the system-level backup that was dumped to tape. You can also use the RECOVER RESTOREBEFORE option to direct the utility to use a recovery base prior to the system-level backup.

**Related reference:**
Chapter 41, “DSNJU004 (print log map),” on page 927

---

**Recovering a table space**

Each table space that is involved is unavailable for most other applications until recovery is complete. If you make image copies by table space, you can recover the entire table space, or you can recover a data set or partition from the table space. If you make image copies separately by partition or data set, you must recover the partitions or data sets by running separate RECOVER operations.

The following RECOVER statement specifies that the utility is to recover table space DSN8S11D in database DSN8D11A:

```
RECOVER TABLESPACE DSN8D11A.DSN8S11D
```

To recover multiple table spaces, create a list of table spaces that are to be recovered; repeat the TABLESPACE keyword before each specified table space. The following RECOVER statement specifies that the utility is to recover partition 2 of the partitioned table space DSN8D11A.DSN8S11E, and recover the table space DSN8D11A.DSN8S11D to the quiesce point (RBA X’000007425468’).

```
RECOVER TABLESPACE DSN8D11A.DSN8S11E DSNUM 2
    TABLESPACE DSN8D11A.DSN8S11D
    TORBA X’000007425468’
```

The following example shows the RECOVER statement for recovering four data sets in database DSN8D11A, table space DSN8S11E:

```
RECOVER PARALLEL (4)
    TABLESPACE DSN8D11A.DSN8S11E DSNUM 1
    TABLESPACE DSN8D11A.DSN8S11E DSNUM 2
    TABLESPACE DSN8D11A.DSN8S11E DSNUM 3
    TABLESPACE DSN8D11A.DSN8S11E DSNUM 4
```

Each of the 4 partitions will be restored in parallel. You can also schedule the recovery of these data sets to run in four separate jobs.

If a table space or data set is in the COPY-pending status, recovering it might not be possible.

**Related concepts:**
"Resetting COPY-pending status” on page 356
Recovering a list of objects

You can recover table spaces, table space partitions, pieces of a nonpartitioned table space, index spaces, index space partitions, and indexes.

When you recover an object to a prior point in time, you should recover a set of referentially related table spaces together to avoid putting any of the table spaces in CHECK-pending status. Use REPORT TABLESPACESET to obtain a table space listing.

Objects that are to be restored from a system-level backup are restored by the main task for the RECOVER utility by invoking DFSMShsm. Objects that are to be restored from a FlashCopy image copy are restored by invoking DFSMShsm.

Each object can have a different base from which to recover: system-level backup, image copy, concurrent copy, or FlashCopy image copy.

RECOVER does not place dependent table spaces that are related by informational referential constraints into CHECK-pending status.

The RECOVER utility merges incremental copies serially and dynamically. As a result, recovery of a table space list with numerous incremental copies can be time-consuming and operator-intensive.

If referential integrity violations are not an issue, you can run a separate job to recover each table space.

When you specify the PARALLEL keyword, Db2 supports parallelism during the RESTORE phase and performs recovery as follows:

- During initialization and setup (the UTILINIT recover phase), the utility locates the full and incremental copy information for each object in the list from SYSIBM.SYSCOPY.
- The utility sorts the list of objects for recovery into lists to be processed in parallel according to the number of tape volumes, file sequence numbers, and sizes of each image copy.
- The number of objects that can be restored in parallel depends on the maximum number of available tape devices and on how many tape devices the utility requires for the incremental and full image copy data sets. You can control the number of objects that are to be processed in parallel on the PARALLEL keyword. You can control the number of dynamically allocated tape drives on the TAPEUNITS keyword, which is specified with the PARALLEL keyword.
- If an object in the list requires a Db2 concurrent copy, the utility sorts the object in its own list and processes the list in the main task, while the objects in the other sorted lists are restored in parallel. If the concurrent copies that are to be restored are on tape volumes, the utility uses one tape device and counts it toward the maximum value that is specified for TAPEUNITS.
- If objects in the list require a system-level backup that was dumped to tape as its recovery base (that is, the FROMDUMP option was specified), the RECOVER utility will invoke DFSMShsm to restore the data sets for the objects in parallel. In this case, the degree of parallelism cannot exceed the maximum number of tasks that can be started by the RECOVER utility. DFSMShsm restores the data sets in parallel based on its installation options.
Recovering a data set or partition

You can use the RECOVER utility to recover individual partitions and data sets. The phases for data set recovery are the same as for table space recovery.

About this task

If image copies are taken at the data set level, RECOVER must be run at the data set level.

If you took FlashCopy image copies at the table space level (by specifying the DSNUM ALL option in the COPY statement), you do not have to recover all of the data sets individually. Even though SYSIBM.SYSCOPY contains records for each partition or piece of the FlashCopy image copy data set, you can recover the entire table space as one object in the RECOVER statement. This recovery is possible, because SYSIBM.SYSCOPY also contains a record to indicate that the FlashCopy image copy was taken at the table space level. Alternatively, you can recover each of the data sets individually.

Restriction: RECOVER does not support recovery of the following data sets and partitions:
- A single data set for nonpartitioned secondary indexes
- A logical partition of a nonpartitioned secondary index

Procedure

To recover a data set or partition, take one of the following actions:
- If image copies are taken at the data set level and you want to recover the whole table space, recover all of the data sets individually in one or more RECOVER steps. If recovery is attempted at the table space level, Db2 returns an error message.
- If image copies are taken at the table space, index, or index space level, recover individual data sets by using the DSNUM parameter in the RECOVER statement.

Recovery with incremental copies

The RECOVER utility attempts to use all incremental image copies that were taken since the last full copy.

RECOVER tries to merge all of these incremental image copies into one copy. To do this merging, all of these copies must be available at the same time. In some cases, this requirement can strain your system resources, for example, by demanding more tape units than are available. However, even if all of these copies cannot be made available at the same time, the utility still uses whichever incremental image copies it can.

Recommendation: If your system resources might be strained by making all required incremental image copies available, consider running MERGECOPY regularly to merge incremental image copies into one copy.

During recovery, the utility dynamically allocates the full image copy and then attempts to dynamically allocate all required incremental image copy data sets. If RECOVER successfully allocates every incremental copy, recovery proceeds to merge pages to table spaces and apply the log. If a point is reached where an...
incremental copy cannot be allocated, RECOVER notes the log RBA or LRSN of the last successfully allocated data set. RECOVER stops trying to allocate any more incremental image copy data sets and merges only those data sets that have already been allocated. RECOVER then applies the log from the noted RBA or LRSN, and ignores the incremental image copies that were not allocated.

---

**Recovering with FlashCopy image copies**

Recovering from a FlashCopy image copy is potentially faster than recovering from a traditional image copy. If an appropriate FlashCopy image copy is available, the RECOVER utility can use it to instantaneously restore an image copy.

**About this task**

Consider the following information when planning for recovery with FlashCopy image copies:

- Create both FlashCopy image copies and traditional sequential image copies to provide a complete recovery base for media failures.
- Creating and recovering from a consistent FlashCopy image copy can consume more processing resources than creating and recovering from a FlashCopy image copy that was not guaranteed to be consistent. To recover from a consistent FlashCopy image copy, the RECOVER utility must read the logs to apply changes that were made to the recovered object after the point of consistency. Some of those changes are likely to be work that was previously backed out and must be reapplied, because the work was uncommitted at the time the FlashCopy image copy was created. (In this case, recovery requires more analysis of the logs during the PRELOGC phase. The preliminary log apply phase, PRELOGA, and the other log phases also require more analysis.) The START_RBA value in the SYSCOPY record of the FlashCopy image copy indicates the low RBA or LRSN of the logs that are needed for processing by RECOVER.
- Recovering with FlashCopy image copies could prevent a subsequent BACKUP SYSTEM utility job on the same data from completing successfully if the FlashCopy relationship is still outstanding. This limitation exists because of the characteristics of FlashCopy relationships. When the RECOVER utility uses fast replication to restore a FlashCopy image copy, it establishes a FlashCopy relationship. This relationship is between the FlashCopy image copy data set (the FlashCopy source) and the underlying data sets for the table space or index space (the FlashCopy target). Cascading FlashCopy relationships, where a data set or extent is both a FlashCopy target and a source, is not supported by the hardware. BACKUP SYSTEM also uses FlashCopy technology. Therefore, any of the FlashCopy targets from the RECOVER operation (those underlying data sets for the table space or index space) cannot also be used as a source for BACKUP SYSTEM while the FlashCopy relationship from RECOVER still exists.
- If you plan to use BACKUP SYSTEM, use the REC_FASTREPLICATION subsystem parameter as described in step 1 to control whether the RECOVER utility should use FlashCopy to restore FlashCopy image copies.
- If the FlashCopy image copy has been migrated or deleted, RECOVER uses the equivalent sequential image copies (created from the FlashCopy image copy) if any exist. Otherwise, the next most recent eligible recovery base is used.
Procedure

To recover with FlashCopy image copies:

1. Ensure that the REC_FASTREPLICATION subsystem parameter is set to PREFERRED or REQUIRED.
   If this subsystem parameter is set to PREFERRED, RECOVER attempts to use fast replication (FlashCopy) to restore the FlashCopy image copy. Fast replication is not used if the underlying data sets for the table space or index space are already in a FlashCopy relationship due to the BACKUP SYSTEM utility or to the COPY utility. In this case, traditional I/O methods are used instead of fast replication to restore the FlashCopy image copy.
   If this subsystem parameter is set to REQUIRED, RECOVER must use fast replication to restore the FlashCopy image copy. If fast replication cannot be used, the recovery of the object fails. For example, the recovery could fail if the BACKUP SYSTEM utility has established a FlashCopy relationship with the production volume.

   **Restriction:** If the RECOVER utility establishes a FlashCopy relationship to restore a FlashCopy image copy and the BACKUP SYSTEM utility is started, the creation of the system-level backup might fail. The reason for the failure is because cascading FlashCopy relationships are not supported.
   Otherwise, if this subsystem parameter is set to NONE, RECOVER restores the FlashCopy image copy using traditional I/O methods. Use this option if you use BACKUP SYSTEM and you do not want recovery from FlashCopy image copies to interfere with the creation of system-level backups.

2. Specify an appropriate RECOVER utility control statement. You do not need to specify any extra options in the RECOVER statement to indicate that you want FlashCopy image copies to be used. The RECOVER utility uses FlashCopy image copies if available.
   However, if you specify that RECOVER is to use a specific FlashCopy image copy (by specifying TOCOPY, TOLASTCOPY, or TOLASTFULLCOPY), specify the DSNUM option with the appropriate data set or partition number for the FlashCopy image copy that you want to use.
   If you took FlashCopy image copies at the table space level (by specifying the DSNUM ALL option in the COPY statement), you do not have to recover all of the data sets individually. You can recover the entire table space as one object in the RECOVER statement. Alternatively, you can recover each of the data sets individually.

**Related concepts:**
- “FlashCopy image copies” on page 157
- “Subsystem parameters for refining DFSMSdss COPY operation with utilities” on page 40

**Related reference:**
- Encrypted FlashCopy image copies, copies made with DFSMSdss concurrent copy, and system-level backups (Managing Security)

**Recovering a page**

Using RECOVER PAGE enables you to recover data on a page that is damaged.
In some situations, you can determine (usually from an error message) which page of an object has been damaged. You can use the PAGE option to recover a single page. You can use the CONTINUE option to continue recovering a page that was damaged during the LOGAPPLY phase of a RECOVER operation.

Suppose that you start RECOVER for table space TSPACE1. During processing, message DSNl0121 informs you of a problem that damages page number 5. RECOVER completes, but the damaged page, number 5, is in a stopped state and is not recovered. When RECOVER ends, message DSNU5011 informs you that page 5 is damaged.

To repair the damaged page:

1. Use the DUMP option of the REPAIR utility to view the contents of the damaged page. Determine what change should have been made by the applicable log record, and apply it by using the REPLACE option of REPAIR. Use the RESET option to turn off the inconsistent-data indicator.

   **Attention:** Be extremely careful when using the REPAIR utility to replace data. Using REPAIR to change data to invalid values can produce unpredictable results, particularly when you change page header information. Improper use of REPAIR can result in damaged data, or in some cases, system failure.

2. Resubmit the RECOVER utility job by specifying TABLESPACE(TSPACE1) PAGE(5) CONTINUE. The RECOVER utility finishes recovering the damaged page by applying the log records that remain after the one that caused the problem.

If more than one page is damaged during RECOVER, perform the preceding steps for each damaged page.

### Recovering an error range

By using the ERROR RANGE option of RECOVER, you can repair pages with reported I/O errors. Db2 maintains a page error range for I/O errors for each data set; pages within the range cannot be accessed. The DISPLAY DATABASE command displays the range.

When recovering an error range, RECOVER:

1. Locates, allocates, and applies image copies.
2. Applies changes from the log.

The following RECOVER statement specifies that the utility is to recover any current error range problems for table space TS1:

```
RECOVER TABLESPACE DB1.TS1 ERROR RANGE
```

Recovering an error range is useful when the range is small, relative to the object containing it; otherwise, recovering the entire object is preferable.

Message DSNU086I indicates that I/O errors were detected on a table space and that you need to recover it. Before you attempt to use the ERROR RANGE option of RECOVER, you should run the ICKDSF service utility to correct the disk error. If an I/O error is detected during RECOVER processing, Db2 issues message DSNU5381 to identify the affected target tracks are involved. The message provides enough information to run ICKDSF correctly.
In some situations, which are announced by error messages, recovery of only an error range is not possible. In such a situation, recovering the entire object is preferable.

During the recovery of the entire table space or index space, Db2 might still encounter I/O errors that indicate Db2 is still using a bad volume. For user-defined data sets, you should use Access Method Services to delete the data sets and redefine them with the same name on a new volume. If you use Db2 storage groups, you can remove the bad volume from the storage group by using ALTER STOGROUP. If you use DFSMS storage groups, you should also remove the bad volume from the DFSMS storage group.

---

**Effect on RECOVER of the NOT LOGGED or LOGGED table space attributes**

You can recover NOT LOGGED table spaces to any recoverable point.

Recoverable points are established when you take one of the following actions:
- Alter a table space from LOGGED to NOT LOGGED. If a base table space is altered to NOT LOGGED and its associated LOB table spaces already have the NOT LOGGED attribute, the ALTER to NOT LOGGED is not a recoverable point.
- Take an image copy from a NOT LOGGED table space.
- When a table has the NOT LOGGED attribute, and an ALTER TABLE with the ADD PARTITION clause is executed.
- When insertion of data into a partition-by-growth table space causes Db2 to add a new partition.

To recover a set of objects with LOB relationships, you should run RECOVER with the TOLOGPOINT option to identify a common recoverable point for all objects. For a non-LOB table space, or a LOB table space with a base table space that has the NOT LOGGED attribute, the logging attribute of the table space must meet these following conditions:
- For recovery to the current point in time, the current value of the logging attribute of the object must match the logging attribute at the most current recoverable point.
- For recovery to a prior point in time, the current value of the logging attribute of the object must match the logging attribute at the time that is specified by TOLOGPOINT, TORBA, TOCOPY, TOLASTCOPY, or TOLASTFULLCOPY

---

**Recovering with a data set copy that is not made by Db2**

You can restore a data set to a point of consistency by using a data set copy that was not made by the COPY utility.

After recovery to the point of consistency, if you choose to continue and recover to the current point in time, you do not want RECOVER to begin processing by restoring the data set from a Db2 image copy. Therefore, use the LOGONLY option of RECOVER, which causes RECOVER to skip the RESTORE phase and apply the log records only, starting from the first log record that was written after the data set was backed up.
Because the data sets are restored offline without Db2 involvement, RECOVER LOGONLY checks that the data set identifiers match those that are in the Db2 catalog. If the identifiers do not match, message DSNU548I is issued, and the job terminates with return code 8.

You can use the LOGONLY option on a list of objects.

To ensure that no other transactions can access Db2 objects between the time that you restore a data set and the time that you run RECOVER LOGONLY, follow these steps:

1. Stop the Db2 objects that are being recovered by issuing the following command:
   ```
   -STOP DATABASE(database-name) SPACENAM(space-name)
   ```

2. Restore all Db2 data sets that are being recovered.

3. Start the Db2 objects that are being recovered by issuing the following command:
   ```
   -START DATABASE(database-name) SPACENAM(space-name) ACCESS(UT)
   ```

4. Run the RECOVER utility without the TORBA or TOLOGPOINT parameters and with the LOGONLY parameter to recover the Db2 data sets to the current point in time and to perform forward recovery using Db2 logs. If you want to recover the Db2 data sets to a prior point in time, run the RECOVER utility with either TORBA or TOLOGPOINT, and with the LOGONLY parameters.

5. If you did not recover related indexes in the same RECOVER control statement, rebuild all indexes on the recovered object.

6. Issue the following command to allow access to the recovered object if the recovery completes successfully:
   ```
   -START DATABASE(database-name) SPACENAM(space-name) ACCESS(RW)
   ```

With the LOGONLY option, when recovering a single piece of a multi-piece linear page set, RECOVER opens the first piece of the page set. If the data set is migrated by DFSMShsm, the data set is recalled by DFSMShsm. Without LOGONLY, no data set recall is requested.

Backing up a single piece of a multi-piece linear page set is not recommended. This action can cause a data integrity problem if the backup is used to restore the data set at a later time.

---

**Recovering catalog and directory objects**

If you need to recover the catalog and directory, you must recover them before you recover user table spaces. Also, you must recover catalog and directory objects in a specific order.

**Before you begin**

Before you can recover catalog and directory objects, you must meet the following requirements:

- Ensure that you have applied APAR PI11316 and completed the ++HOLD actions.
- Converting page sets to or from extended 10-byte format during REBUILD INDEX on catalog and directory indexes complicates recovery and might cause failures. To avoid conversions, set the UTILITY_OBJECT_CONVERSION...
subsystem parameter to NONE until all catalog and directory objects are recovered. Alternatively, specify RBALRSN_CONVERSION NONE on all REBUILD INDEX steps.

- Recovering and rebuilding catalog and directory objects requires installation SYSADM or installation SYSOPR authority.
- If you are performing a recovery at a remote site, start the remote Db2 for z/OS subsystem with ACCESS(MAINT) specified on the START DB2 command and with DEFER ALL specified in the DSNZPARM load module. (See installation panels DSNTIPS and DSNTIPO3.) If Db2 is not started with ACCESS(MAINT), resource unavailable conditions on the real-time statistics (RTS) catalog indexes might occur during REBUILD INDEX(ALL) for the catalog and directory indexes.
- If the logging environment requires adding or restoring active logs, restoring archive logs, or performing any action that affects the log inventory in the BSDS, you need to recover the BSDS before catalog and directory objects. To copy active log data sets, use the Access Method Services REPRO function.

A full recovery of the catalog and directory is recommended. However, if you need to do a point-time-recovery, be aware of the implications associated with doing point-in-time recovery of the catalog, directory, and user objects. See “Point-in-time recovery of the catalog, directory, and all user objects” on page 492.

Tip: For best results, check the consistency of the Db2 catalog and directory regularly, even outside of the migration process. For detailed instructions, see Verify the integrity of Db2 table spaces and Check for consistency between catalog tables.

About this task

The following table spaces do not have entries in SYSIBM.SYSLGRNX. The indexes that are associated with these table spaces also do not have entries in SYSIBM.SYSLGRNX, even if they were defined with COPY YES. These objects are assumed to be open from the point of their last image copy, so the RECOVER utility processes the log from that point forward.

- DSNDB01.SYSUTILX
- DSNDB01.DB01
- DSNDB01.SYSLGRNX
- DSNDB06.SYSTSCPY (SYSTSCPY replaces SYSCOPY during enabling-new-function mode processing. The SYSCOPY catalog table space is used in conversion mode until the SYSCOPY table space is replaced during enabling-new-function mode.)
- DSNDB01.SYSDBDXA

Requirement: You must recover the catalog and directory objects in the order that is specified in this task. If you are recovering any subset of the objects in the list, start with the object that is listed first and continue in the order of the list. For example, if you need to recover SYSLGRNX, SYSUTILX, and SYSUSER, recover SYSUTILX first, then SYSLGRNX, and then SYSUSER. You do not need to recover all of the objects; only recover those objects that require recovery.

General guidelines: Use the following general guidelines for all of the steps in this task:

- If you copy your catalog or directory indexes, use the RECOVER utility to recover your indexes. Otherwise, use the REBUILD INDEX utility to rebuild those indexes.
• For all catalog and directory table spaces, you can list the IBM-defined indexes that have the COPY YES attribute in the same RECOVER utility statement.

• Recovery of the items in the list can be done concurrently or included in the same job step. However, the following restrictions apply:
  – When you recover the following table spaces or indexes, the job step in which the RECOVER statement appears must not contain any other utility statements. No other utilities can run while the RECOVER utility is running.
    - DSNDB01.SYSUTILX
    - All indexes on SYSUTILX
    - DSNDB01.DBD01
  – When you recover the following table spaces, no other utilities can run while the RECOVER utility is running. Other utility statements can exist in the same job step.
    - DSNDB06.SYSTSCPY
    - DSNDB01.SYSLGRNX

Special instructions for conversion mode: If the Db2 for z/OS subsystem is in conversion mode, recover the catalog and directory objects in the order that is defined for Version 10 (DB2 10 for z/OS: Recovering catalog and directory objects) with one change. In step “Recover all of the remaining catalog and directory table spaces,” add the following objects:

Table 66. Additional table spaces to recover in conversion mode (CM10)

<table>
<thead>
<tr>
<th>Additional catalog table spaces to recover in conversion mode (CM10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSTSOBX</td>
</tr>
<tr>
<td>SYSTSQRE</td>
</tr>
<tr>
<td>SYSTSQRS</td>
</tr>
<tr>
<td>SYSTSVAR</td>
</tr>
<tr>
<td>SYSTSVAT</td>
</tr>
</tbody>
</table>

Procedure

To recover catalog and directory objects:

1. Recover DSNDB01.SYSUTILX.
2. Run REBUILD INDEX(ALL) on DSNDB01.SYSUTILX.
3. Recover DSNDB01.DBD01.
4. Run REBUILD INDEX(ALL) on DSNDB01.DBD01.
5. Recover DSNDB01.SYSDBDXA.
6. Run REBUILD INDEX(ALL) on DSNDB01.SYSDBDXA.
7. Recover DSNDB06.SYSTSCPY.
   If the SYSCOPY table space has not been converted by enabling-new-function mode processing (job DSNTIJEN step ENFM0037), recover DSNDB06.SYSTSCPY instead of DSNDB06.SYSTSCPY
8. Run REBUILD INDEX(ALL) on DSNDB06.SYSTSCPY.
   If the SYSCOPY table space has not been converted by enabling-new-function mode processing (job DSNTIJEN step ENFM0037), run REBUILD INDEX(ALL) on DSNDB06.SYSTSCPY instead of DSNDB06.SYSTSCPY.
9. Recover DSNDB01.SYSLGRNX.
10. Run REBUILD INDEX(ALL) on DSNDB01.SYSLGRNX.
11. Recover DSNDB06.SYSTSSTG and DSNDB06.SYSTSVOL.
12. Run REBUILD INDEX(ALL) on DSNDB06.SYSTSSTG and DSNDB06.SYSTSVOL.

13. Recover all of the remaining catalog and directory table spaces in a single RECOVER utility statement with the PARALLEL option.
   The remaining catalog and directory table spaces include the following table spaces:
   - The directory table spaces in DSNDB01:
     - SCT02
     - SPT01
     - SYSSPUXA
     - SYSSPUXB
   - The catalog table spaces that are listed in the following table:

   **Table 67. Remaining catalog table spaces to recover in new-function mode**

<table>
<thead>
<tr>
<th>Table</th>
<th>SYSLALTER</th>
<th>SYSTSCON</th>
<th>SYSTSPKC</th>
<th>SYSTSTSM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>SYSCONTX</td>
<td>SYSTSCDT</td>
<td>SYSTSPKD</td>
<td>SYSTSSYN</td>
</tr>
<tr>
<td>Table</td>
<td>SYSDDF</td>
<td>SYSTSCTL</td>
<td>SYSTSPKG</td>
<td>SYSTSTAB</td>
</tr>
<tr>
<td>Table</td>
<td>SYSEBCDC</td>
<td>SYSTSCTR</td>
<td>SYSTSPKL</td>
<td>SYSTSTAU</td>
</tr>
<tr>
<td>Table</td>
<td>SYSGPAUT</td>
<td>SYSTSDAT</td>
<td>SYSTSPKS</td>
<td>SYSTSTBC</td>
</tr>
<tr>
<td>Table</td>
<td>SYSGRTN5</td>
<td>SYSTSDBA</td>
<td>SYSTSPKX</td>
<td>SYSTSTPF</td>
</tr>
<tr>
<td>Table</td>
<td>SYSHIST</td>
<td>SYSTSDBR</td>
<td>SYSTSPKY</td>
<td>SYSTSTPT</td>
</tr>
<tr>
<td>Table</td>
<td>SYSJAUXA</td>
<td>SYSTDBU</td>
<td>SYSTSPA</td>
<td>SYSTSTRG</td>
</tr>
<tr>
<td>Table</td>
<td>SYSJAUXB</td>
<td>SYSTDEP</td>
<td>SYSTSPLD</td>
<td>SYSTSTRT</td>
</tr>
<tr>
<td>Table</td>
<td>SYSJAVA</td>
<td>SYSTSENV</td>
<td>SYSTSPLN</td>
<td>SYSTSP</td>
</tr>
<tr>
<td>Table</td>
<td>SYSPLUXA</td>
<td>SYSTSFUA</td>
<td>SYSTSPLY</td>
<td>SYSTSS</td>
</tr>
<tr>
<td>Table</td>
<td>SYSPLUXB</td>
<td>SYSTSFLD</td>
<td>SYSTPRH</td>
<td>SYSTSV</td>
</tr>
<tr>
<td>Table</td>
<td>SYSROLES</td>
<td>SYSTSFOR</td>
<td>SYSTSPRM</td>
<td>SYSTSVAD</td>
</tr>
<tr>
<td>Table</td>
<td>SYSEQ</td>
<td>SYSTSIPT</td>
<td>SYSTSPX</td>
<td>SYSTSV</td>
</tr>
<tr>
<td>Table</td>
<td>SYSEQ2</td>
<td>SYSTSISS</td>
<td>SYSTSPR</td>
<td>SYSTSVAT</td>
</tr>
<tr>
<td>Table</td>
<td>SYSTATS</td>
<td>SYSTSIXC</td>
<td>SYSTSQA</td>
<td>SYSTSVAU</td>
</tr>
<tr>
<td>Table</td>
<td>SYSTARG</td>
<td>SYSTSIXR</td>
<td>SYSTSQE</td>
<td>SYSTSVW</td>
</tr>
<tr>
<td>Table</td>
<td>SYSTSADT</td>
<td>SYSTSIXS</td>
<td>SYSTSORO</td>
<td>SYSTSVT</td>
</tr>
<tr>
<td>Table</td>
<td>SYSTSASC</td>
<td>SYSTSIXT</td>
<td>SYSTSORP</td>
<td>SYSTSVWD</td>
</tr>
<tr>
<td>Table</td>
<td>SYSTSATS</td>
<td>SYSTSKY</td>
<td>SYSTSOR</td>
<td>SYSTSVT</td>
</tr>
<tr>
<td>Table</td>
<td>SYSTSATW</td>
<td>SYSTSKYC</td>
<td>SYSTSQRY</td>
<td>SYSTSTXM</td>
</tr>
<tr>
<td>Table</td>
<td>SYSTSATX</td>
<td>SYSTSOBX</td>
<td>SYSTSAU</td>
<td>SYSTTS</td>
</tr>
<tr>
<td>Table</td>
<td>SYSTSAUX</td>
<td>SYSTSPDO</td>
<td>SYSTSREL</td>
<td>SYUSER</td>
</tr>
<tr>
<td>Table</td>
<td>SYSTSCHX</td>
<td>SYSTSPDT</td>
<td>SYSTSOU</td>
<td>SYXML</td>
</tr>
<tr>
<td>Table</td>
<td>SYSTSCKD</td>
<td>SYSTSPEN</td>
<td>SYSTSSCM</td>
<td></td>
</tr>
<tr>
<td>Table</td>
<td>SYSTSCKS</td>
<td>SYSTSPHX</td>
<td>SYSTSSF</td>
<td></td>
</tr>
<tr>
<td>Table</td>
<td>SYSTSCOL</td>
<td>SYSTSPKA</td>
<td>SYSTSSR</td>
<td></td>
</tr>
</tbody>
</table>

   - If DSNDB06.SYRTSTS has not been converted by enabling-new-function mode processing (job DSNTIJEN, step ENFM0047), run REBUILD INDEX(ALL) on DSNDB06.SYRTSTS instead of DSNDB06.SYSTSISS and DSNDB06.SYSTSTSS.

   - If DSNDB06.SYSTSTR has not been converted by enabling-new-function mode processing (job DSNTIJEN, step ENFM0077), run REBUILD INDEX(ALL) on DSNDB06.SYSTSTR instead of DSNDB06.SYSCON and DSNDB06.SYSTSTSS.

14. Run REBUILD INDEX(ALL) on DSNDB06.SYSTSISS and DSNDB06.SYRTSTS.
   If the SYRTSTS table space has not been converted by enabling-new-function mode processing (job DSNTIJEN steps ENFM0047), run REBUILD INDEX(ALL) on DSNDB06.SYRTSTS instead of DSNDB06.SYSTSISS and DSNDB06.SYRTSTS.
15. Run REBUILD INDEX(ALL) on DSNDB06.SYSTSTSP, DSNDB06.SYSTSTPT, and DSNDB06.SYSTSTAB.

   Execute the following utility statements to rebuild the IBM-defined and any user-defined indexes on the table spaces:

   ```sql
   REBUILD INDEX (ALL) TABLESPACE DSNDB06.SYSTSTSP
   RBALRSN_CONVERSION NONE
   REBUILD INDEX (ALL) TABLESPACE DSNDB06.SYSTSTPT
   RBALRSN_CONVERSION NONE
   REBUILD INDEX (ALL) TABLESPACE DSNDB06.SYSTSTAB
   RBALRSN_CONVERSION NONE
   ```

16. Rebuild all of the remaining catalog and directory indexes using REBUILD INDEX(ALL) for each table space recovered in step 13 on page 488. Rebuild all user-defined indexes on the catalog that have not been rebuilt or recovered yet.

**Results**

**Messages that you might receive:**

**DSNU1530I**

If you are recovering the catalog and directory during migration to a new version, you might receive message DSNU1530I with RC0. This message is issued if your RECOVER statement or REBUILD INDEX statement specifies a catalog or directory object that no longer exists in the new version or has not yet been created in the new version. In this case, Db2 skips the object. No action is required.

**DSNT500I**

An ID with a granted authority receives message DSNT500I RESOURCE UNAVAILABLE while trying to recover a table space in the catalog or directory if certain table spaces in the following list are unavailable:

- DSNDB06.SYSTSF
- DSNDB06.SYSTSCOL
- DSNDB06.SYSTSTSP
- DSNDB06.SYSTSTPT
- DSNDB06.SYSTSTAB
- DSNDB06.SYSTSIXS
- DSNDB06.SYSTSIXT
- DSNDB06.SYSTSIXR
- DSNDB06.SYSTSIPT
- DSNDB06.SYSTSREL
- DSNDB06.SYSTSFOR
- DSNDB06.SYSTSSYN
- DSNDB06.SYSTSFLD
- DSNDB06.SYSTSTA
- DSNDB06.SYSTSKEY
- DSNDB06.SYSUSER

If you receive message DSNT500I, you must take one of the following actions:

- Make these table spaces available.
• Run the RECOVER utility on the catalog or directory by using an authorization ID that has the installation SYSADM or installation SYSOPR authority.

**DSNT501I**

If certain table spaces in DSNDB06 are unavailable when an ID with a granted authority tries to rebuild indexes in the catalog or directory, Db2 issues message DSNT501I.

**What to do next**

After you recover the Db2 catalog and directory, perform the following actions:

• After a point-in-time recovery on the catalog and directory, run the CHECK DATA utility on the objects to ensure consistency.

• Recover XML scheme repository objects. Although the XML schema repository database, DSNXSR, is not part of the Db2 catalog, you need to recover all table spaces in the DSNXSR database and rebuild all indexes on those table spaces immediately after you recover the Db2 catalog. If you perform a point-in-time recovery of the catalog, you need to recover objects in the DSNXSR database to the same point in time.

Related concepts:

- Management of the bootstrap data set (Db2 Administration Guide)

Related reference:

- Db2 catalog tables (Db2 SQL)
- Db2 directory tables (Db2 SQL)
- Administrative authorities (Managing Security)
- -START DB2 (Db2) (Db2 Commands)
- DSNTIPO3: Default startup modules panel (Db2 Installation and Migration)
- DSNTIPS: Databases and spaces to start automatically panel (Db2 Installation and Migration)

“Syntax and options of the CHECK DATA control statement” on page 74
“Syntax and options of the OPTIONS control statement” on page 407
“Syntax and options of the REBUILD INDEX control statement” on page 428
“Syntax and options of the RECOVER control statement” on page 461

Related information:

- REPRO command (DFSMS Access Method Services for Catalogs)

**Objects that contain recovery information**

To recover one object, the RECOVER utility must obtain information about it from another object. Therefore, dependencies exist between catalog and directory objects, and you must recover them in a specific order.

The following table lists the objects from which RECOVER must obtain information.
Table 68. Objects that the RECOVER utility accesses

<table>
<thead>
<tr>
<th>Object name</th>
<th>Reason for access by RECOVER</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNDB01.SYSUTILX</td>
<td>Utility restart information. The object is not accessed when it is recovered; a RECOVER job for this object is not restartable, and no other commands can be in the same job step. SYSCOPY information for SYSUTILX is obtained from the log. You can use REPORT RECOVERY to obtain SYSCOPY information for DSNDB01.SYSUTILX.</td>
</tr>
<tr>
<td>DSNDB01.DB01, DSNDB01.SYSDBDXA</td>
<td>Descriptors for the catalog database (DSNDB06), the work file database (DSNDB07), and user databases. A RECOVER job for this object is not restartable, and no other commands can be in the same job step. SYSCOPY information for DBD01 and SYSDBDXA is obtained from the log. You can use REPORT RECOVERY to obtain SYSCOPY information for DSNDB01.DB01 and DSNDB01.SYSDBDXA.</td>
</tr>
<tr>
<td>DSNDB06.SYSTSCPY</td>
<td>Locations of image copy data sets. This table space contains the SYSIBM.SYSCOPY table. SYSCOPY information for SYSTSCPY itself is obtained from the log. You can use REPORT RECOVERY to obtain SYSCOPY information for DSNDB06.SYSTSCPY.</td>
</tr>
<tr>
<td>DSNDB01.SYSLGRNX</td>
<td>The RBA or LRSN of the first log record after the most recent copy.</td>
</tr>
<tr>
<td>DSNDB06.SYSTSDBA, DSNDB06.SYSTSDBU, DSNDB06.SYSUSER</td>
<td>Verification that the authorization ID is authorized to run RECOVER.</td>
</tr>
<tr>
<td>From several of the following table spaces in DSNDB06:</td>
<td>Information about table spaces that are to be recovered.</td>
</tr>
<tr>
<td>• SYSTSFAU</td>
<td></td>
</tr>
<tr>
<td>• SYSTSCOL</td>
<td></td>
</tr>
<tr>
<td>• SYSTSTSP</td>
<td></td>
</tr>
<tr>
<td>• SYSTSTPT</td>
<td></td>
</tr>
<tr>
<td>• SYSTSTAB</td>
<td></td>
</tr>
<tr>
<td>• SYSTSIXS</td>
<td></td>
</tr>
<tr>
<td>• SYSTSTIXT</td>
<td></td>
</tr>
<tr>
<td>• SYSTSTIXR</td>
<td></td>
</tr>
<tr>
<td>• SYSTSIPT</td>
<td></td>
</tr>
<tr>
<td>• SYSTSREL</td>
<td></td>
</tr>
<tr>
<td>• SYSTSFOR</td>
<td></td>
</tr>
<tr>
<td>• SYSTSSYN</td>
<td></td>
</tr>
<tr>
<td>• SYSTSFLD</td>
<td></td>
</tr>
<tr>
<td>• SYSTSTAU</td>
<td></td>
</tr>
<tr>
<td>• SYSTSKEY</td>
<td></td>
</tr>
</tbody>
</table>
Point-in-time recovery of the catalog, directory, and all user objects

Full recovery of the catalog and directory table spaces and indexes is strongly recommended. However, in some situations, you might need to do a point-in-time recovery. In this case, you should understand the implications and plan for this type of recovery.

When you recover the Db2 catalog, directory, and all user objects, consider the entire catalog and directory, including all table spaces and index spaces, to be one logical unit. Recover all objects in the catalog, directory, and all user objects to the same point of consistency. If you plan to do a point-in-time recovery of the catalog, directory, and all user objects, a separate quiesce of the DSNDB06.SYSTSCPY table space is required after a quiesce of the other catalog and directory table spaces.

The enabling-new-function mode process changes catalog and directory objects from BASIC format to EXTENDED format. Point-in-time recoveries of these objects need to be recovered in the correct format. Systems that are recovered during conversion mode must be recovered to a point-in-time when the catalog and directory were in BASIC mode. After the BSDS is converted to extended mode, the catalog and directory must be recovered in EXTENDED mode. During the enabling-new-function mode process and up to the conversion of the BSDS, catalog objects might exist in both BASIC and EXTENDED format.

A point-in-time recovery on catalog and directory objects bypasses the checking for the following items:
- A complete referential integrity (RI) set. If the complete RI set is not recovered together, CHKP is not set on the dependents.
- A complete base and LOB set. If base and LOB objects are not recovered together, ACHKP or CHKP is not set.

**Recommendation:** Before you recover the Db2 catalog, directory, and all user objects to a prior point in time, shut down the Db2 subsystem cleanly and then restart the subsystem in ACCESS(MAINT) mode. Recover the catalog and directory objects to the point in time. You can use sample queries and documentation, which are provided in DSNTESQ in the SDSNSAMP sample library, to check the consistency of the catalog.

If you perform a point-in-time recovery on catalog and directory table spaces, the indexes are placed in RBDP (rebuild-pending) status. Use the CHECK INDEX utility to determine whether an index is inconsistent with the data that it indexes. You can use the REBUILD INDEX utility to rebuild the indexes. Alternatively, you can use the RECOVER utility to recover catalog and directory indexes if the index was defined with the COPY YES attribute and if you have a full index image copy.
Creating a point of consistency for catalog and directory objects

Full recovery of the catalog and directory table spaces and indexes is strongly recommended. However, if you need to plan for point-in-time recovery of the catalog and directory, you should create a point of consistency for the catalog and directory.

Procedure

To create a point of consistency for catalog and directory objects:

1. Quiesce all catalog and directory table spaces, except for DSNDB06.SYSTSCPY and DSNDB01.SYSUTILX.
   You can use the LISTDEF utility to group these table spaces into a single list and then specify that list in the QUIESCE statement.

2. Quiesce DSNDB06.SYSTSCPY.
   **Recommendation:** Quiesce DSNDB06.SYSTSCPY in a separate utility statement. When you recover DSNDB06.SYSTSCPY to its own quiesce point, it contains the SYSCOPY records with ICTYPE = 'Q' (quiesce) for the other catalog and directory table spaces.

3. Quiesce DSNDB01.SYSUTILX in a separate job step.

What to do next

Later, if you need to recover to a point in time, recover DSNDB06.SYSTSCPY and DSNDB01.SYSUTILX to their own quiesce points, and recover other catalog and directory table spaces to their common quiesce point. The catalog and directory objects must be recovered in the prescribed order for your version of Db2 for z/OS.

Related tasks:

"Recovering catalog and directory objects” on page 485

Related reference:

- [Db2 catalog tables (Db2 SQL)](#)
- [Db2 directory tables (Db2 SQL)](#)
- “Syntax and options of the QUIESCE control statement” on page 416
- [SYSIBM.SYSCOPY table (Db2 SQL)](#)
- [SYSIBM.SYSUTILX table (Db2 SQL)](#)
- Chapter 18, “LISTDEF,” on page 215

Reinitializing DSNDB01.SYSUTILX

You need to reinitialize the DSNDB01.SYSUTILX directory table space if you cannot successfully execute the DISPLAY UTILITY and TERMINATE UTILITY commands. In this case, DSNDB01.SYSUTILX is damaged and you cannot recover DSNDB01.SYSUTILX, because errors occur in the LOGAPPLY phase.
About this task

Because DSNDDB01.SYSUTILX contains information about active and outstanding utilities, the process of reinitializing this table space involves determining which objects have a utility in progress and resolving any pending states to make the object available for access.

Procedure

If DSNDDB01.SYSUTILX must be reinitialized, use the following procedure with caution:

1. Issue the **-DIS DB(*) SPACENAM(*) RESTRICT** command and analyze the output. Write down the following items:
   - All of the objects with a utility in progress (The objects in UTUT, UTRO, or UTRW status have utilities in progress.)
   - Any pending states for these objects (RECP, CHKP, and COPY are examples of pending states.)

2. Run the following utility statement to reset the SYSUTILX table space:

   ```
   CATMAINT UPDATE UTILX RESET
   Initialize SYSUTILX and its indexes.
   ```

3. Issue the **-START DB(dbname) ACCESS(UT)** command for each database that has objects with a utility in progress.

4. Issue the **-START DB(dbname)SPACENAM(spname) ACCESS(FORCE)** command on each object with a utility in progress. This action clears all utilities that are in progress or in pending states. (Any pending states are cleared, but you still need to resolve the pending states as directed in the next step.)

5. Resolve the pending states for each object by running the appropriate utility. For example, if an object was in the RECP status, run the RECOVER utility.

6. Issue **-START DB(dbname) ACCESS(RW)** for each database.

Related reference:

[Appendix B, “Advisory or restrictive states,” on page 1037](#)

---

Recovering a table space that contains LOB or XML data

The RECOVER utility can set the auxiliary warning status for a LOB table space or XML table space if it finds at least one invalid LOB or XML column.

Db2 marks a LOB or XML column invalid if all of the following conditions are true:

- The LOB table space or XML table space was defined with the LOG(NO) attribute.
- The LOB table space or XML table space was recovered.
- The LOB or XML was updated since the last image copy.

The status of an object that is related to a LOB or XML table space can change due to a recovery operation, depending on the type of recovery that is performed. If all of the following objects for all LOB or XML columns are recovered in a single RECOVER utility statement to the present point in time, no pending status exists:

- Base table space
- Index on the auxiliary table
- LOB table space
- XML table space
The RECOVER utility verifies that all related objects that are required to perform a point-in-time recovery are included in the RECOVER control statement. The VERIFYSET keyword enables you to control whether a point-in-time recovery requires all base, LOB, XML, and history objects in a set. You can choose to break up point-in-time recoveries into multiple jobs with VERIFYSET NO.

Related reference:
Appendix B, “Advisory or restrictive states,” on page 1037

Recovering a table space that contains clone objects

The recovery guidelines and considerations for a cloned table space or cloned index are the same as for a base table space or base index except in the point-in-time recovery case.

For an object currently involved in cloning, or one that was previously involved in cloning, a point-in-time recovery cannot be done to a time the precedes the most recent EXCHANGE statement. The time of the most recent EXCHANGE for a table space can be determined by querying SYSIBM.COPY for the table space to be recovered where ICTYPE = 'A' and STYPE = 'E'.

When an EXCHANGE is done, two rows will be written to SYSIBM.SYSCOPY for the table space being processed: one for the base object and one for the clone object. These rows are differentiated by the SYSCOPY.INSTANCE column value: one will have INSTANCE=1 and the other INSTANCE=2. These SYSIBM.SYSCOPY rows do not indicate base or clone. The SYSIBM.SYSCOPY.INSTANCE column value can be used to determine which SYSIBM.SYSCOPY row is for a base object and which is for a clone object.

Point-in-time recovery

Recovering data to a prior time is a point-in-time recovery. You can recover objects to a particular RBA, LRSN, or image copy. You can do this type of recovery by using the RECOVER utility point-in-time recovery options. These options are TOCOPY, TOLOGPOINT, TOLASTCOPY, TORBA, and TOLASTFULLCOPY.

You can recover objects to any RBA or LRSN by using TORBA or TOLOGPOINT.
You can recover objects to a previous image copy by using TOCOPY, TOLASTCOPY, or TOLASTFULLCOPY.

When you recover objects to an RBA or LRSN, the RBA or LRSN does not have to be a consistent point in time. The RECOVER utility automatically handles any uncommitted units of work and the data is left in a consistent state.

When you recover objects to an image copy, whether the image copy is a consistent point in time depends on the type of image copy. An image copy that was taken with SHRLEVEL REFERENCE is a point of consistency. An image copy that was taken with SHRLEVEL CHANGE is not an explicit point of consistency.

Another explicit point of consistency is a quiesce point, which is a point at which data is consistent as a result of running the Db2 QUIESCE utility.

Recoveries to a consistent point in time are the most efficient because no uncommitted units of work need to be backed out.
**Recommendation:** If you use the RECOVER utility to recover data to an image copy by specifying TOCOPY, TOLASTCOPY, or TOLASTFULLCOPY, specify a copy that was made with the SHRLEVEL REFERENCE option.

To achieve consistency when you want to recover to a copy that was taken with SHRLEVEL CHANGE, specify a recovery point immediately after the copy completed. To find this point, locate a record for the SHRLEVEL CHANGE copy in SYSIBM.SYSCOPY and use the value in the PIT_RBA column. Specify that recovery point by using the TORBA or TOLOGPOINT options in the RECOVER statement.

You do not need to take a full image copy after you recover data to a point in time, except in the case of fallback recovery. Db2 records the RBAs or LRSNs that are associated with the point-in-time recovery in the SYSIBM.SYSCOPY catalog table to allow future recover operations to skip the unwanted range of log records.

If you specify the TOCOPY, TOLASTCOPY, or TOLASTFULLCOPY option to recover data to a point in time, RECOVER puts any associated index spaces in REBUILD-pending status. If you specify the TOLOGPOINT or TORBA option to recover data to a point in time, RECOVER puts any associated index spaces in REBUILD-pending status if the indexes are not recovered in the same RECOVER statement as their corresponding table space. The reason is that a point-in-time recovery of only the table space leaves data in a consistent state and indexes in an inconsistent state.

You can remove the REBUILD-pending state in one of the following ways:
- Run REBUILD INDEX on the indexes.
- Run RECOVER to a point in time on the indexes. If you do that, Db2 sets the CHECK-pending state on the indexes, because the table space was not recovered in the same RECOVER utility statement as the indexes.

If you use a point-in-time recovery option to recover a single data set of a nonpartitioned table space, Db2 issues message DSNU520I to warn that the table space can become inconsistent following the RECOVER job. This point-in-time recovery can cause compressed data to exist without a dictionary or can even overwrite the data set that contains the current dictionary.

If you use the point-in-time recovery option to recover a partition-by-growth table space that has an image copy with fewer partitions than the current table space, any excess partitions (partitions that are currently defined but not in the image copy) are empty after the RECOVER processing.

If a table space or partition in reordered row format is recovered to a point in time when the table space or partition was in basic row format, the table space or partition reverts to basic row format after RECOVER processing. Similarly, if a table space or partition in basic row format is recovered to a point in time when the table space or partition was in reordered row format, the table space or partition revert to reordered row format after RECOVER processing.

After recovering a set of table spaces to a point in time, you can use CHECK DATA to check for inconsistencies.

If you use the RECOVER utility to recover a table space set to a point-in-time, you must ensure that you recover the entire set of table spaces to the same point in
time. If you do not include every member of the set, or if you do not recover the entire set to the same point in time, RECOVER sets the auxiliary CHECK-pending status on for all table spaces in the set.

You can also use point-in-time recovery and the point-in-time recovery options to recover all user-defined table spaces and indexes that are in refresh-pending status (REFP).

**Recommendation:** After running any point-in-time recoveries, run REORG TABLESPACE and REBUILD INDEX to set the real-time statistics. For more information about the effect of point-in-time recoveries on real-time statistics, see "Effects of running RECOVER" on page 510.

**Requirement:** To use system-level backups as a recovery base, DFSMShsm must be at z/OS 1.8 or higher.

### Backing out work to a point-in-time

The RECOVER utility can recover your data to a point in time by backing out committed work from the current state of the data. To recover data by backing out, specify BACKOUT YES on the RECOVER control statement.

In some circumstances, recovering to a point in time by backing out work can be faster than recovering to a point in time by restoring a copy of the data and applying the logs forward.

When the RECOVER utility performs a point-in-time recovery by backing out committed work, the recovery is a point-in-time recovery with consistency, because any work that was uncommitted at the point in time to which the data is being recovered is also backed out. When the recovery is complete, the data is left in a transaction consistent state.

**Restrictions:** You cannot perform a backout recovery to the following points in time:

- A point-in-time before a utility that inserts SYSCOPY records was run, unless the utility is COPY or COPYTOCOPY.
- A point-in-time before REORG TABLESPACE with the LOG(YES) option was run on the table space.

Before running the RECOVERY utility with the BACKOUT YES option, run the REPORT utility with the RECOVER option on the object being recovered to identify events that might prevent you from recovering the object by backing out work to a given point in time.

### Recovery considerations after rebalancing partitions with REORG

For partitioned table spaces, image copies that were taken before a REORG job that materialized limit key changes are not usable for recovering to a current RBA or LRSN. Avoid recovering a partitioned table space to a point-in-time that is after the
REORG-pending or advisory REORG-pending status was set but before the REORG that redistributed data records. To determine an appropriate point in time:
1. Run REPORT RECOVERY.
2. Select an image copy for which the recovery point is a point after the REORG that redistributed data records.

Suppose that you run the REORG utility to turn off a REORG-pending status, and then recover to a point in time before that REORG job. In this case, Db2 sets restrictive statuses on all partitions that you specified in the REORG job, as follows:
- Sets REORG-pending (and possibly CHECK-pending) on for the data partitions
- Sets REBUILD-pending on for the associated index partitions
- Sets REBUILD-pending on for the associated logical partitions of nonpartitioned secondary indexes

To create a new consistent recovery point, take one of the following actions immediately after an ALTER INDEX, ALTER TABLE, or REORG REBALANCE operation that changes partition boundaries:
- Run REORG with the COPYDDN and SHRLEVEL NONE options.
- Take a full image copy immediately after REORG completes.

Using offline copies to recover after rebalancing partitions

To recover data after a REORG job redistributes the data among partitions, use RECOVER LOGONLY. If you perform a point-in-time recovery, you must keep the offline copies synchronized with the SYSCOPY records. Therefore, do not use the MODIFY RECOVERY utility to delete any SYSCOPY records with an ICTYPE column value of ‘A’ because these records might be needed during the recovery. Delete these SYSCOPY records only when you are sure that you no longer need to use the offline copies that were taken before the REORG that performed the rebalancing.

Restrictions for point-in-time recoveries

The following restrictions apply to point-in-time recoveries:
- You can take system-level backups with the BACKUP SYSTEM utility. However, if any of the following utilities were run since the system-level backup that was chosen as the recovery base, then the use of the system-level backup is prohibited for object level recoveries to a prior point in time:
  - REORG TABLESPACE
  - REORG INDEX
  - REBUILD INDEX
  - LOAD REPLACE
  - RECOVER from image copy or concurrent copy
  This restriction does not apply if you are using z/OS V1R11.0 or later and you set up DFSMShsm to capture catalog information.
- RECOVER cannot recover an index to a point in time if pending definition changes on that object were materialized by the REORG utility after that point in time.
- You cannot use RECOVER to a point in time on an index to reset the REBUILD-pending state unless the index is in the REBUILD-pending state because the associated table space was recovered to a point in time, and no pending definition change is involved.
• RECOVER cannot recover a partition-by-growth table space to a point-in-time before REORG TABLESPACE removed empty, trailing partitions.

• RECOVER cannot recover an XML table space to a point-in-time before the REORG utility that changed the format from basic to extended format.

• The following restrictions apply to recovery to a point in time that is before materialization of pending definition changes.
  – The target object must be a partition-by-range table space, a LOB table space, or an XML table space.
  – For most types of pending changes, the target object must be an entire table space. This restriction does not apply for pending changes to partition limit keys.
  – The table space cannot be recovered with VERIFYSET NO. If VERIFYSET NO is specified, RECOVER uses VERIFYSET YES instead.
  – The RECOVER statement cannot specify TOCOPY, TOLASTCOPY, or TOLASTFULLCOPY if the source for recovery is an image copy that was taken with the SHRLEVEL CHANGE option.
  – The target table space cannot contain a clone table.
    If a clone table exists in the table space, but no data exchange occurred, you can drop the clone table, and then perform the point-in-time recovery. If data exchange occurred, you cannot perform the point-in-time recovery.
  – The pending definition change cannot be:
    - A change to the table space type (with an ALTER TABLESPACE statement)
    - A change to hash organization (with an ALTER TABLE statement and an ALTER ORGANIZATION clause)
    - A change to drop a column (with an ALTER TABLE statement and a DROP COLUMN clause)
  – You cannot run RECOVER jobs that recover the table space to a different point in until you run the REORG utility to resolve the REORG-pending (REORP) state on the table space. However, this restriction does not apply for pending definition changes to limit keys of partitions.

• For an object currently involved in cloning, or one that was previously involved in cloning, a point-in-time recovery cannot be done to a time the precedes the most recent EXCHANGE statement.

**Actions that can affect recovery status**

When you perform the following actions before you recover a table space, the recovery status is affected as described:

• If you alter a table to rotate a partition (with an ALTER TABLE statement and a ROTATE PARTITION clause):
  – You can recover the partition to the current time.
  – You can recover the partition to a point in time after the alter. The utility can use a recovery base, (for example, a full image copy, a REORG LOG YES operation, or a LOAD REPLACE LOG YES operation) that occurred before the alter.
  – You cannot recover the partition to a point in time before the alter; the recover fails with MSGDSNU556I and RC8.

• If you change partition boundaries (with an ALTER TABLE statement and an ALTER PARTITION clause or with a REORG REBALANCE utility control statement):
You can recover the partition to the current time if a recovery base (for example, a full image copy, a REORG LOG YES operation, or a LOAD REPLACE LOG YES operation) exists.

- You can recover the partition to a point in time after the change.
- You can recover the partitions that are affected by the boundary change to a point in time before the materialization of those changes by the REORG TABLESPACE utility. However, after the RECOVER utility completes successfully, the affected partitions with the limit key changes are placed in REORG-pending (REORP) status. You then need to run REORG TABLESPACE to correctly redistribute the data according to the previous limit key values.

- If you alter a table to add a partition (with an ALTER TABLE statement and an ADD PARTITION clause):
  - You can recover the partition to the current time.
  - You can recover the partition to a point in time after the alter.
  - You can recover the partition to a point in time before the alter; RECOVER resets the partition to be empty.

- If you add a column (with an ALTER TABLE statement and an ADD COLUMN clause), you cannot recover a table space to a point in time between the time that you alter the table to add a column and the time that you take either of the following actions:
  - Drop the default value (with an ALTER TABLE statement and an ALTER COLUMN clause that specifies DROP DEFAULT)
  - Alter the default value (with an ALTER TABLE statement and an ALTER COLUMN clause that specifies SET DEFAULT)

- If you convert a table to support multiple XML versions (with a REORG TABLESPACE utility control statement):
  - You cannot recover the associated table space to a point in time before the table was converted.
  - You cannot recover any indexes for that table to a point in time before the table was converted.

- If you alter the organization of your table space to hash organization (with an ALTER TABLE statement and an ALTER ORGANIZATION clause):
  - You can recover the table space to the current time.
  - You can recover the table space to a point in time before or after the alter.
  - You can recover the table space to a point in time before or after the REORG that materialized the hash organization. RECOVER places the table space in AREOR status if the table space was recovered to a point before the REORG.

- If you alter the size of the hash space in your table space (with an ALTER TABLE statement and an ALTER ORGANIZATION clause):
  - You can recover the table space to the current time.
  - You can recover the table space to a point in time before or after the alter.
  - You can recover the table space to a point in time before or after the REORG that materialized the change in hash space size.

- If you drop the hash organization (with an ALTER TABLE statement and a DROP ORGANIZATION clause):
  - You can recover the table space to the current time.
  - You can recover the table space to a point in time after the alter.
  - You cannot recover the table space to a point in time before the alter.

- If you execute pending definition changes, you cannot perform a point-in-time recovery until you have either materialized the pending definition changes (with a REORG TABLESPACE or REORG INDEX utility control statement) or dropped those changes (with an ALTER TABLESPACE statement and a DROP PENDING CHANGES clause).

Examples: Changing the following characteristics results in pending definition changes:
– The segment size (with an ALTER TABLESPACE statement and a SEGSIZE clause)
– The data set size (with an ALTER TABLESPACE statement and a DSSIZE clause)
– The buffer pool page size (with an ALTER TABLESPACE statement and a BUFFERPOOL clause)
– The MEMBER CLUSTER attribute (with an ALTER TABLESPACE statement and a MEMBER CLUSTER clause)
– The table space type (with an ALTER TABLESPACE statement)
– The limit key values (with an ALTER TABLE statement and an ALTER PARTITION clause)

• If you perform any of the following SQL operations on a table in a segmented table space or universal table space, you cannot back out the changes (with a RECOVER utility control statement and a BACKOUT YES clause):
  – DELETE without a WHERE clause (mass DELETE)
  – TRUNCATE TABLE
  – DROP TABLE
  – ALTER TABLE with a ROTATE PARTITION clause

If you perform any of the previously indicated actions on tables in a base table space that has indexes or auxiliary objects (LOB tables spaces or XML table spaces), this restriction also applies to those indexes or auxiliary objects.

When you perform the following actions before you recover an index to a prior point in time or to the current time, the recovery status is affected as described:
• If you alter the data type of a column to a numeric data type (with an ALTER TABLE statement and an ALTER COLUMN clause specifying the new data type), you cannot recover the index until you take a full image copy of the index. However, the index can be rebuilt.
• If you alter an index to NOT Padded or Padded (with an ALTER INDEX statement and a NOT Padded or Padded clause), you cannot recover the index until you take a full image copy of the index. However, the index can be rebuilt.
• If you regenerate an index (with an ALTER INDEX statement and a REGENERATE clause), you cannot recover the index or index space to a point in time prior to the time that it was regenerated. Instead, rebuild the index by using the REBUILD INDEX utility.
• If you alter an index such that Db2 creates a new version of the index, you cannot recover the index to a point in time prior to the first ALTER INDEX statement that created a new version of that index.

Planning for point-in-time recovery

Recovering to a point in time that is a point of consistency (QUIESCE or SHRLEVEL REFERENCE set) is desirable because there will be no uncommitted work to back out.

When making copies of a single object, use SHRLEVEL REFERENCE to establish consistent points for TOCOPY, TOLASTCOPY, or TOLASTFULLCOPY recovery. Copies that are made with SHRLEVEL CHANGE do not copy data at a single instant because changes can occur as the copy is made. A subsequent RECOVER TOCOPY operation can produce inconsistent data. Instead use RECOVER with the TOLOGPOINT option to identify a point after the SHRLEVEL CHANGE copy and any uncommitted units of work will be backed out.

When copying a list of objects, use SHRLEVEL REFERENCE. If a subsequent recovery to a point in time is necessary, you can use a single RECOVER utility
statement to list all of the objects, along with TOLOGPOINT to identify the common RBA or LRSN value. If you use SHRLEVEL CHANGE to copy a list of objects, you should follow it with a QUIESCE of the objects.

To improve the performance of the recovery, take a full image copy of the table space or set of table spaces, and then quiesce them by using the QUIESCE utility. This action enables RECOVER TORBA or TOLOGPOINT to recover the table spaces to the quiesce point with minimal use of the log.

Authorization: Restrict use of the point-in-time recovery options to personnel with a thorough knowledge of the Db2 recovery environment.

Ensuring consistency

You can use RECOVER TORBA, RECOVER TOLOGPOINT, and RECOVER TOCOPY to recover one of the following single objects:
• Partition of a partitioned table space
• Partition of a partitioning index space
• Data set of a simple table space

For any of the previously listed objects, restore all data sets to the same level; otherwise, the data becomes inconsistent.

If possible, specify a table space and all of its indexes (or a set of table spaces and all related indexes) in the same RECOVER utility statement, and specify TOLOGPOINT or TORBA to identify a QUIESCE point. This action avoids placing indexes in the CHECK-pending or REBUILD-pending status. If the TOLOGPOINT is not a common QUIESCE point for all objects, use the following procedure:
1. RECOVER table spaces to the value for TOLOGPOINT (either an RBA or LRSN).
2. Use concurrent REBUILD INDEX jobs to recover the indexes over each table space.

This procedure ensures that the table spaces and indexes are synchronized, and it eliminates the need to run the CHECK INDEX utility.

If you cannot specify TOLOGPOINT or TORBA to identify a QUIESCE point, you can specify any point in time, and Db2 will leave the data in a consistent state. The RECOVER utility automatically handles any uncommitted units of work and leaves the data in a consistent state when TORBA or TOLOGPOINT is specified.

When using RECOVER with the TORBA or TOLOGPOINT option, ensure that all of the objects that are changed by the active units of recovery at the recovery point are recovered to the same point-in-time so that they are synchronized:
• Db2 rolls back changes made to units of recovery that are inflight, inabort, postponed abort, or indoubt during the recovery point-in-time.
• Db2 does not roll back changes made to units of recovery that are INCOMMIT during the recovery point-in-time.
• Db2 rolls back only changes to objects in the RECOVER statement.

Avoiding CHECK-pending status

Db2 sets CHECK-pending status in the following point-in-time-recovery situations:
• You recover at least one member of a table space set to a prior point in time, but you do not recover all members of the table space set to the same quiesce point. In this case, all dependent table spaces that are recovered are placed in CHECK-pending status with the scope of the whole table space. All dependent table spaces of the recovered table spaces are placed in CHECK-pending status with the scope of the specific dependent tables.

• The RECOVER statement contains the TORBA option or TOLOGPOINT option and recovers all members of a table space set to the same point in time. However, referential constraints were defined in one of those table spaces after that point in time. In this case, the CHECK-pending status is set for the table space that contains the table with the referential constraint.

• The RECOVER statement contains the TORBA option or TOLOGPOINT option and recovers one or more indexes to a previous point in time. However, the same RECOVER statement does not recover the related table space. In this case, Db2 sets the CHECK-pending status for the indexes.

RECOVER does not place dependent table spaces that are related by informational referential constraints into CHECK-pending status.

To avoid setting CHECK-pending status, take the following actions:

• When you recover tables that are involved in a referential constraint, recover all of the table spaces that are involved in the constraint.

• Recover all dependent objects to the same point in time.

• Do not add table check constraints or referential constraints after the point in time to which you want to recover.

• Recover indexes and the related table space to the same point in time (preferably a quiesce point) or COPY SHRLEVEL REFERENCE point. RECOVER processing resets the CHECK-pending status for all indexes in the same RECOVER statement.

Related information:
- Recovery of table space sets (Db2 Administration Guide)
- “CHECK-pending status” on page 1039

Compressed data

Use caution when recovering a portion of a table space or partition (for example, one data set) to a prior point in time. If the data set that is being recovered has been compressed with a different dictionary, you can no longer read the data.

Recovery to a point in time before materialization of pending definition changes

You can recover a partition-by-range table space, a LOB table space, or an XML table space to a point in time before a REORG job was run to materialize pending definition changes. However, the REORG job that materialized the changes must have been run in Db2 11 new-function mode.

Restriction: For a list of restrictions on recovery to a point in time before materialization of pending definition changes, see Chapter 26, “RECOVER,” on page 459.

Before you run RECOVER to a point in time that is before materialization of pending definition changes, run REPORT RECOVERY to obtain:
• The recovery history from the SYSIBM.SYSCOPY catalog table
• The log ranges from the SYSIBM.SYSLGRNX directory table

After you run RECOVER to a point in time that is before materialization of pending definition changes, the target table space is put in the REORG-pending state. You must run REORG on the entire table space to remove the REORG-pending state and complete the recovery process.

Related concepts:
“How the RECOVER utility performs fallback recovery” on page 508
“Recovery of data to a prior point in time (Db2 Administration Guide)”
“Resetting the REBUILD-pending status” on page 452

Related tasks:
“Compressing your data (Db2 Performance)”
“Materializing pending definition changes (Db2 Administration Guide)”
“Reviewing CHECK INDEX output” on page 117

Related reference:
Appendix B, “Advisory or restrictive states,” on page 1037
“REBUILD-pending status” on page 1043
“Syntax and options of the RECOVER control statement” on page 461
“REORG-pending status” on page 1045

Avoiding specific image copy data sets during a recovery

You might accidentally lose an image copy, or you might want to avoid a specific image copy data set. Because the corresponding row is still present in SYSIBM.SYSCOPY, the RECOVER utility always attempts to allocate the data set.

Use the RESTOREBEFORE option and specify the RBA or LRSN of the image copy, concurrent copy, or system-level backup that you want to avoid, and RECOVER will search for an older recovery base. The RECOVER utility then applies log records to restore the object to its current state or the specified TORBA or TOLOGPOINT value.

Image copy on tape

If the image copy is on tape, messages IEF233D and IEF455D request the tape for RECOVER, as shown in the following example:

```
  IEF233D M BAB,COPY ,R92341QJ,DSNUPROC,
  OR RESPOND TO IEF455D MESSAGE
*42 IEF455D MOUNT COPY ON BAB FOR R92341QJ,DSNUPROC OR REPLY 'NO'
  R 42,NO
  IEF234E K BAB,COPY ,PVT,R92341QJ,DSNUPROC
```

By replying NO, you can initiate the fallback to the previous image copy. RECOVER responds with messages DSNU030I and DSNU508I, as shown in the following example:

```
DSNU030I  csect-name - UNABLE TO ALLOCATE R92341Q.UTQPS001.FCOPY010
          RC=4, CODE=X'04840000'
DSNU508I  csect-name - IN FALBACK PROCESSING TO PRIOR FULL IMAGE COPY
```

Reason code X'0484' means that the request was denied by the operator.
Image copy on disk:

If the image copy is on disk, you can delete or rename the image copy data set before RECOVER starts executing. RECOVER issues messages DSNU030I and DSNU508I, as shown in the following example:

DSNU030I csect-name - UNABLE TO ALLOCATE R92341Q.UQPS001.FCOPY010, RC=4, CODE=X'17080000'
DSNU508I csect-name - IN FALBACK PROCESSING TO PRIOR FULL IMAGE COPY

Reason code X'1708' means that the ICF catalog entry cannot be found.

How to improve RECOVER performance

You can improve the performance of the RECOVER utility by taking certain actions.

To improve recovery time, consider recovering to a quiesce point or SHRLEVEL REFERENCE copy instead of recovering to any point in time. The following factors impact performance when you recover to a non quiesce point:

• The duration of the units of recovery that were active at the recovery point.
• The number of Db2 members that have active units of recovery to roll back.

Use MERGECOPY to merge your table space image copies before recovering the table space. If you do not merge your image copies, RECOVER automatically merges them. If RECOVER cannot allocate all the incremental image copy data sets when it merges the image copies, RECOVER uses the log instead.

Include a list of table spaces and indexes in your RECOVER utility statement to apply logs in a single scan of the logs.

If you use RECOVER TOCOPY for full image copies, you can improve performance by using data compression. The improvement is proportional to the degree of compression.

Consider specifying the PARALLEL keyword to restore image copies from disk or tape to a list of objects in parallel.

If you are recovering concurrent copies, consider specifying the CURRENTCOPYONLY option to improve performance. When you specify this option, RECOVER can issue one DFSMSdss RESTORE command for multiple objects. The utility issues one RESTORE command for each group of objects that is associated with the concurrent copy data set. If you do not use the CURRENTCOPYONLY keyword, RECOVER issues one RESTORE command for each object.

If you are recovering an object from a system-level backup, RECOVER invokes DFSMSshm, which controls parallelism. If the system-level backup resides on disk, the RECOVER utility passes the object to DFSMSshm before processing the objects to be restored from image copies or concurrent copies. If the system-level backup resides on tape, the RECOVER utility processes the objects to be restored from system-level backups, image copies, and concurrent copies at the same time.

Recovery from a FlashCopy image copy with consistency or from a sequential image copy with consistency might take longer due to the additional processing required to read the logs and apply any changes made after the point of consistency.
Optimizing the LOGAPPLY phase

The time that is required to recover a table space depends also on the time that is required to read and apply log data. You can take several steps to optimize the process. If possible, Db2 reads the required log records from the active log to provide the best performance.

Any log records that are not found in the active logs are read from the archive log data sets, which are dynamically allocated to satisfy the requests. The type of storage that is used for archive log data sets is a significant factor in the performance. Consider the following actions to improve performance:

- RECOVER a list of objects in one utility statement to take only a single pass of the log.
- Keep archive logs on disk to provide the best possible performance.
- Control archive logs data sets by using DFSMShsm to provide the next best performance. Db2 optimizes recall of the data sets. After the data set is recalled, Db2 reads it from disk.
- If the archive log must be read from tape, Db2 optimizes access by means of ready-to-process and look-ahead mount requests. Db2 also permits delaying the deallocation of a tape drive if subsequent RECOVER jobs require the same archive log tape. Those methods are described in more detail in the subsequent paragraphs.

The BSDS contains information about which log data sets to use and where they reside. You must keep the BSDS information current. If the archive log data sets are cataloged, the ICF catalog indicates where to allocate the required data set.

DFSMShsm data sets

The recall of the first DFSMShsm archive log data set starts automatically when the LOGAPPLY phase starts. When the recall is complete and the first log record is read, the recall for the next archive log data set starts. This process is known as look-ahead recalling. Its purpose is to recall the next data set while it reads the preceding one.

When a recall is complete, the data set is available to all RECOVER jobs that require it. Reading proceeds in parallel.

Non-DFSMShsm tape data sets

Db2 reports on the console all tape volumes that are required for the entire job. The report distinguishes two types of volumes:

- Any volume that is not marked with an asterisk (*) is required for the job to complete. Obtain these volumes from the tape library as soon as possible.
- Any volume that is marked with an asterisk (*) contains data that is also contained in one of the active log data sets. The volume might or might not be required.

As tapes are mounted and read, Db2 makes two types of mount requests:

- Ready-to-process: The current job needs this tape immediately. As soon as the tape is loaded, Db2 allocates and opens it.
- Look-ahead: This is the next tape volume that is required by the current job. Responding to this request enables Db2 to allocate and open the data set before it is needed, thus reducing overall elapsed time for the job.
You can dynamically change the maximum number of input tape units that are used to read the archive log by specifying the COUNT option of the SET ARCHIVE command. For example, use the following command to assign 10 tape units to your Db2 subsystem:

```
-SET ARCHIVE COUNT (10)
```

The DISPLAY ARCHIVE READ command shows the currently mounted tape volumes and their statuses.

**Delayed deallocation**

Db2 can delay deallocating the tape units used to read the archive logs. This is useful when several RECOVER utility statements run in parallel. By delaying deallocation, Db2 can re-read the same volume on the same tape unit for different RECOVER jobs, without taking time to allocate it again.

You can dynamically change the amount of time that Db2 delays deallocation by using the TIME option of the SET ARCHIVE command. For example, to specify a 60 minute delay, issue the following command:

```
-SET ARCHIVE TIME(60)
```

In a data sharing environment, you might want to specify zero (0) to avoid having one member hold onto a data set that another member needs for recovery.

**Performance summary**

1. Achieve the best performance by allocating archive logs on disk.
2. Consider staging cataloged tape data sets to disk before allocation by the log read process.
3. If the data sets are read from tape, set both the COUNT and the TIME values to the maximum allowable values within the system constraints.

---

**Recovering image copies in a JES3 environment**

You can recover sequential or concurrent image copies in a JES3 environment.

**Procedure**

To recover image copies in a JES3 environment:

Ensure that sufficient units are available to mount the required image copies. In a JES3 environment, if the number of image copies that need to be restored exceeds the number of available online and offline units, and the RECOVER job successfully allocates all available units, the job waits for more units to become available.

---

**Resetting RECOVER-pending or REBUILD-pending status**

Several possible operations on a table space can place the table space in the RECOVER-pending status and the index space in REBUILD-pending status.

**Procedure**

To reset RECOVER-pending or REBUILD-pending status:

Use one of the following methods:
• Recover the table space, index space, or partition.
• Use REBUILD INDEX to rebuild the index space from existing data.
• Use the LOAD utility, with the REPLACE option, on the table space or partition.
• Use the REPAIR utility, with the NORCVRPEND option, on the table space, index space, or partition. Be aware that the REPAIR utility does not fix the data inconsistency in the table space or index.
• Rebuild indexes, run REORG TABLESPACE SORTDATA for table spaces and indexes.

Related reference:
“REBUILD-pending status” on page 1043
“RECOVER-pending status” on page 1044
Chapter 25, “REBUILD INDEX,” on page 427
Chapter 28, “REORG TABLESPACE,” on page 553
Chapter 29, “REPAIR,” on page 669

How the RECOVER utility performs fallback recovery

The RECOVER utility attempts to use the latest primary copy data set as a starting point for recovery. If the latest primary copy data set is not available, RECOVER attempts to use the backup copy data set, if one is available.

If neither image copy is usable, RECOVER attempts to fall back to a previous recovery point. If the previous recovery point is a full image copy, the RECOVER utility uses the full image copy, any incremental image copies, and the log to recover. If a previous REORG LOG YES or LOAD REPLACE LOG YES was done, RECOVER attempts to recover from the log and applies any changes that occurred between the two image copies. If good full image copies are not available, and no previous REORG LOG YES or LOAD REPLACE LOG YES jobs were run, the RECOVER utility terminates. The RECOVER utility will not fall back to a system-level backup.

If one of the following actions occurs, the index remains untouched, and utility processing terminates with return code 8:
• RECOVER processes an index for which no full copy exists.
• The copy cannot be used because of utility activity that occurred on the index or on its underlying table space,

If you always make multiple image copies, RECOVER should seldom fall back to an earlier point. Instead, RECOVER relies on the backup copy data set if the primary copy data set is unusable.

In a JES3 environment, you can do a fallback recovery by issuing a JES3 cancel, s command at the time the allocation mount message is issued. This action might be necessary if a volume is not available or if you do not want the given volume.

RECOVER does not perform parallel processing for objects that are in backup or fallback recovery. Instead, the utility performs nonparallel image copy allocation processing of the objects. RECOVER defers the processing of objects that require backup or fallback processing until all other objects are recovered, at which time the utility processes the objects one at a time.

Related concepts:
“Preparing for recovery by using the COPY utility” on page 170
How the RECOVER utility retains tape mounts

The RECOVER utility can automatically retain the tape volumes for the input image copies when a list of objects is being recovered.

For input image copies (for the objects being recovered) that are stacked on one or more tape volumes, you do not need to code JCL DD statements to retain the tape volumes on the tape drive. Instead, you can use the PARALLEL and TAPEUNITS keywords. The PARALLEL keyword directs the RECOVER utility to process the objects in parallel. The objects will be sorted based on how the input image copies are stacked on tape to maximize efficiency during the RESTORE phase by retaining the tape volumes on the tape drive and by restoring the input image copies in the right order (by ascending file sequence numbers). The TAPEUNITS keyword will limit the number of tape units (or drives) that the RECOVER utility will use during the RESTORE phase. In special cases, RECOVER cannot retain all of the tape volumes, so the tape volumes may be demounted and deallocated even if the PARALLEL and TAPEUNITS keywords are specified.

Avoiding damaged media

When a media error is detected, Db2 prints a message that indicates the extent of the damage. If an entire volume is bad and storage groups are being used, you must remove the bad volume first. If you don’t remove the volume the RECOVER utility might re-access the damaged media.

Procedure

To avoid damaged media:
1. Use ALTER STOGROUP to remove the bad volume and add another volume. If your data sets are managed by DFSMS storage group, then you need to also remove the bad volume from the DFSMS storage group.
2. Run the RECOVER utility for all objects on that volume.

What to do next

If the RECOVER utility cannot complete because of severe errors that are caused by the damaged media, you might need to use Access Method Services (IDCAMS) with the NOSCRATCH option to delete the cluster for the table space or index. If the table space or index is defined by using STOGROUP, the RECOVER utility automatically redefines the cluster. For user-defined table spaces or indexes, you must redefine the cluster before invoking the RECOVER utility.

Related tasks:
- Altering Db2 storage groups (Db2 Administration Guide)

Related reference:
- ALTER STOGROUP (Db2 SQL)

Termination or restart of RECOVER

You can terminate and restart the RECOVER utility.

Termination

Terminating a RECOVER job with the TERM UTILITY command leaves the table space that is being recovered in RECOVER-pending status, and the index space
that is being recovered in the REBUILD-pending status. If you recover a table space to a previous point in time, its indexes are left in the REBUILD-pending status. The data or index is unavailable until the object is successfully recovered or rebuilt. If the utility fails in the LOGAPPLY, LOGCSR, or LOGUNDO phases, fix the problem that caused the job to stop and restart the job rather than terminate the job. For the rest of objects in the recover job, the RECOVER utility restores the original image copy and repeats the LOGAPPLY, LOGCSR, and LOGUNDO process again for this subset of objects. All the objects being recovered in one recover job will be available to the application at the end of the RECOVER utility, even if some of the objects do not have any active URs operating on them and therefore no rollback is needed for these objects.

**Restart**

You can restart RECOVER from the last commit point (RESTART(CURRENT)) or the beginning of the phase (RESTART(PHASE)). By default, Db2 uses RESTART(CURRENT).

If you attempt to recover multiple objects by using a single RECOVER statement and the utility fails in:

- The RESTORE phase: All objects in the process of being restored are placed in the RECOVER-pending or REBUILD-pending status. The status of the remaining objects is unchanged.
- The LOGAPPLY phase: All objects that are specified in the RECOVER statement are placed in the RECOVER-pending or REBUILD-pending status.

In both cases, you must identify and fix the causes of the failure before performing a current restart.

If RECOVER fails in the LOGCSR phase and you restart the utility, the utility restart behavior is RESTART(PHASE).

If RECOVER fails in the LOGUNDO phase and you restart the utility, the utility repeats the RESTORE, LOGAPPLY, LOGCSR, and LOGUNDO phases for only those objects that had active units of recovery that needed to be handled and that did not complete undo processing prior to the failure.

**Related tasks:**

“Restarting an online utility” on page 44

---

**Effects of running RECOVER**

The effects of running the RECOVER utility vary depending on your situation.

**RECOVER without the REUSE option**

When you run the RECOVER utility without the REUSE option and the data set that contains that data is Db2-managed, Db2 deletes this data set before recovery. Then, Db2 redefines a new data set with a control interval that matches the page size.

**Recovering objects to a previous point in time**

If you use the RECOVER utility to recover objects to a previous point in time, the counter columns in the real-time statistics tables might not be valid. Therefore, after any point-in-time recoveries, you must run the following utilities:
• REORG TABLESPACE to reestablish real-time statistic values for table spaces
• REBUILD INDEX to reestablish real-time statistic values for indexes

These actions do not apply if you recover objects to the current state. When you recover objects to the current state, the counter columns in the real-time statistics tables are still valid. Db2 does not modify them.

Cases when indexes are placed in REBUILD-pending status

When you use the RECOVER utility to recover indexes, an index might be left in REBUILD-pending status. In these rare cases, you must rebuild the index by running the REBUILD INDEX utility.

Indexes are left in REBUILD-pending status, if:
• An index with the COPY YES attribute has gone through the two-pass group buffer pool recovery pending (GRECP) or logical page list (LPL) recovery, and the RECOVER utility needs to apply the logs that are processed by the two-pass LPL or GRECP recovery
• Or the indexes are still in GRECP or LPL status, and the compensation log records are written before the physical undo logs

Sample RECOVER control statements

Use the sample control statements as models for developing your own RECOVER control statements.

Example 1: Recovering a table space

The following control statement specifies that the RECOVER utility is to recover table space DSN8D11A.DSN8S11D to the current point in time.

RECOVER TABLESPACE DSN8D11A.DSN8S11D

Example 2: Recovering a table space partition

The following control statement specifies that the RECOVER utility is to recover the second partition of table space DSN8D11A.DSN8S11D. The partition number is indicated by the DSNUM option.

RECOVER TABLESPACE DSN8D11A.DSN8S11D DSNUM 2

Example 3: Recovering a table space partition to the last image copy that was taken

The following control statement specifies that the RECOVER utility is to recover the first partition of table space DSN8D11A.DSN8S11D to the last image copy that was taken. If the last image copy that was taken is a full image copy, this full image copy is restored. If the last image copy that was taken is an incremental image copy, the most recent full image copy, along with any incremental image copies, are restored.

RECOVER TABLESPACE DSN8D11A.DSN8S11D DSNUM 1 TOLASTCOPY

Example 4: Recovering table spaces to a point in time

The following control statement specifies that the RECOVER utility is to recover the second partition of table space DSN8D11A.DSN8S11E and all of table space
DSN8D11A.DSN8S11D to the indicated quiesce point (LRSN X'00000551BE7D'). The quiesce point is indicated by the TOLOGPOINT option. The value for this option can be either an LRSN or an RBA.

RECOVER TABLESPACE DSN8D11A.DSN8S11E DSNUM 2 TABLESPACE DSN8D11A.DSN8S11D TOLOGPOINT X'00000551BE7D'

Example 5: Recovering an index to the last full image copy that was taken without deleting and redefining the data sets

The following control statement specifies that the RECOVER utility is to recover index ADMF001.IADH082P to the last full image copy. The REUSE option specifies that Db2 is to logically reset and reuse Db2-managed data sets without deleting and redefining them.

RECOVER INDEX ADMF001.IADH082P REUSE TOLASTFULLCOPY

Example 6: Recovering from concurrent copies

The RECOVER utility control statement specifies that the utility is to recover all of the objects that are included in the RCVR4_LIST. This list is defined by the preceding LISTDEF utility control statement. Because the most recent primary copy for all of these objects is a concurrent copy, CURRENTCOPYONLY option in the RECOVER statement improves the performance of restoring these concurrent copies. The LOCALSITE option indicates that RECOVER is to use image copies at the local site.

Example 7: Recovering a list of objects on different tape devices in parallel

The control statement specifies that the RECOVER utility is to recover the list of table spaces. Full image copies and incremental image copies of the eight table spaces are stacked on four different tape volumes. The utility sorts the list of
objects and, if possible, recovers two objects at a time in parallel. This number of objects is specified by the PARALLEL option. The TAPEUNITS option specifies that up to four tape drives are to be dynamically allocated.

Example 8: Recovering a list of objects to a point in time

The following RECOVER control statement specifies that the RECOVER utility is to recover the specified list of objects to a common point in time (LRSN X’00000551BE7D’). The LISTDEF control statement defines which objects are to be included in the list. These objects are logically consistent after successful completion of this RECOVER job. The PARALLEL option indicates that RECOVER is to restore four objects at a time in parallel. If any of the image copies are on tape (either stacked or not stacked), RECOVER determines the number of tape drives to use to optimize the process. Any uncommitted work for all of the objects at the specified RBA are backed out by the recover operation to a point in time with consistency.

```
LISTDEF RCVRLIST INCLUDE TABLESPACE DSN8D81A.DSN8S81D
           INCLUDE INDEX DSN8810.XDEPT1
           INCLUDE INDEX DSN8810.XDEPT2
           INCLUDE INDEX DSN8810.XDEPT3
           INCLUDE TABLESPACE DSN8810.XDEPT1
           INCLUDE INDEX DSN8810.XEMP1
           INCLUDE INDEX DSN8810.XEMP2
RECOVER LIST RCVRLIST TOLOGPOINT X'00000551BE7D' PARALLEL(4)
```

Example 9: Recovering an image copy

The following control statement specifies that RECOVER is to search for an image copy with an RBA or LRSN value earlier than the specified X’00000551BE7D’ value to use in the RESTORE phase. Only specified dumps of the database copy pool are used for the restore of the data sets.

```
RECOVER LIST RCVRLIST RESTOREBEFORE X’00000551BE7D’ PARALLEL(4)
    FROMDUMP DUMPCLASS(dcname)
```

Example 10: Recovering clone table data

The following control statement specifies that the RECOVER utility is to recover only clone table data in DBA90601.TLX9061A and recover the data to the last image copy that was taken. The REUSE option specifies that RECOVER is to logically reset and reuse DB2-managed data sets without deleting and redefining them.

```
RECOVER TABLESPACE DBA90601.TLX9061A REUSE TOLASTCOPY
    CLONE
```
Chapter 27. REORG INDEX

The REORG INDEX online utility reorganizes an index space to improve access performance and reclaim fragmented space. You can specify the degree of access to your data during reorganization, and you can collect inline statistics by using the STATISTICS keyword.

You can determine when to run REORG INDEX by using the LEAFDISTLIMIT catalog query option. If you specify the REPORTONLY option, REORG INDEX produces a report that indicates whether a REORG is recommended; in this case, a REORG is not performed. These options are not available for indexes on the directory.

To avoid the cost of running the RUNSTATS utility afterward, you can also specify the STATISTICS option to collect inline statistics when you run the REORG INDEX utility.

You can use the REORG INDEX utility in conjunction with z/OS DFSMS data set encryption to encrypt or decrypt indexes. REORG INDEX always allocates new Db2-managed data sets unless the REUSE option is specified.

Output

The following list summarizes REORG INDEX output:

REORG INDEX
   Reorganizes the entire index (all parts if partitioning).

REORG INDEX PART n
   Reorganizes PART n of a partitioning index or of a data-partitioned secondary index

Authorization required

To execute this utility, you must use a privilege set that includes one of the following authorities:
  • REORG privilege for the database
  • DBADM or DBCTRL authority for the database. If the object on which the utility operates is in an implicitly created database, DBADM authority on the implicitly created database or DSNDB04 is required.
  • DATAACCESS authority
  • SYSCTRL authority
  • SYSADM authority

To execute this utility on an index space in the catalog or directory, you must use a privilege set that includes one of the following authorities:
  • REORG privilege for the DSNDB06 (catalog) database
  • DBADM or DBCTRL authority for the DSNDB06 (catalog) database.
  • Installation SYSOPR authority
  • SYSCTRL authority
  • SYSADM or Installation SYSADM authority
  • STATS privilege for the database is required if STATISTICS keyword is specified.
While trying to reorganize an index space in the catalog or directory, a user with authority other than installation SYSADM or installation SYSOPR might receive the following message:

DSNT500I "resource unavailable"

This message is issued when the DSNDB06.SYSTSDBA, DSNDB06.SYSTSDBU, or DSNDB06.SYSUSER catalog table space or one of the indexes is unavailable. If this problem occurs, run the REORG INDEX utility again, using an authorization ID with the installation SYSADM or installation SYSOPR authority.

An ID with installation SYSOPR authority can also execute REORG INDEX, but only on an index in the DSNDB06 database.

If either the FLASHCOPY YES or FLASHCOPY CONSISTENT option is specified, the user ID that invokes the REORG INDEX utility must have the authority to execute the DFSMSdss COPY command.

To run REORG INDEX STATISTICS REPORT YES, ensure that the privilege set includes the SELECT privilege on the catalog tables and on the tables for which statistics are to be gathered.

**Execution phases of REORG INDEX**

The REORG INDEX utility operates in these phases:

**UTILINIT**
- Performs initialization and setup

**UNLOAD**
- Unloads index space and writes keys to a sequential data set.

**BUILD**
- Builds indexes. Updates index statistics.

**LOG**
- Processes log iteratively. Used only if you specify SHRLEVEL CHANGE.

**SWITCH**
- Switches access between original and new copy of index space or partition. Used only if you specify SHRLEVEL REFERENCE or CHANGE.

**UTILTERM**
- Performs cleanup. For Db2-managed data sets and either SHRLEVEL CHANGE or SHRLEVEL REFERENCE, the utility deletes the original copy of the table space or index space.

**Related tasks:**
- Maintaining data organization (Db2 Performance)
- Collecting statistics by using Db2 utilities (Db2 Performance)
- Reducing the cost of collecting statistics (Db2 Performance)

**Syntax and options of the REORG INDEX control statement**

The REORG INDEX utility control statement, with its multiple options, defines the function that the utility job performs.
You can create a control statement with the ISPF/PDF edit function. After creating it, save it in a sequential or partitioned data set. When you create the JCL for running the job, use the SYSIN DD statement to specify the name of the data set that contains the utility control statement.

**Syntax diagram**

### Syntax diagram

```
REORG INDEX LIST listdef-name
  index-name-spec [REUSE] [CLONE]
  SHRLEVEL NONE
  [SHRLEVEL REFERENCE deadline-spec drain-spec change-spec]
    [FASTSWITCH YES] [FASTSWITCH NO]
  FORCE NONE
  [FORCE READERS] [FORCE ALL]
  LEAFDISTLIMIT integer
  [REPORTONLY] [UNLOAD CONTINUE]
  UNLOAD PAUSE ONLY
  stats-spec [SORTDEVT device-type] [SORTNUM integer]
  WORKDDN (SYSUT1)
  WORKDDN (ddname)
  FLASHCOPY NO
  [FLASHCOPY YES] [CONSISTENT]
  [FCCOPY DDN (template-name)]
  RBALRSN_CONVERSION NONE
    [BASIC] [EXTENDED]

**Notes:**

1. You cannot use UNLOAD PAUSE with the LIST option.
2. You cannot specify any options in stats-spec with the UNLOAD ONLY option.

### index-name-spec

```
INDEX creator-id. index-name
INDEXSPACE database-name. index-space-name
PART integer
```

Chapter 27. REORG INDEX 517
**deadline-spec**

- **DEADLINE**: NONE
- **DEADLINE**: timestamp
- **DEADLINE**: labeled-duration-expression

**Notes:**
1. The default for DRAIN_WAIT is the value of the IRLMRWT subsystem parameter.
2. The default for RETRY is the value of the UTIMOUT subsystem parameter.
3. The default for RETRY_DELAY is the smaller of the following two values: DRAIN_WAIT value × RETRY value, DRAIN_WAIT value × 10

**change-spec**

- **MAXRO**: integer
- **DRAIN**: ALL
- **DRAIN**: WRITERS
- **LONGLOG**: CONTINUE
- **LONGLOG**: TERM
- **DELAY**: 1200
- **DELAY**: integer
- **LOGRANGES**: YES
- **LOGRANGES**: NO
- **SWGITCHTIME**: NONE
- **SWGITCHTIME**: timestamp
- **Labeled-duratin-expression**:
- **NEWMAXRO**: NONE
- **NEWMAXRO**: integer

**Notes:**
1. The default for MAXRO is the RETRY_DELAY default value.
Notes:
1 The KEYCARD option is deprecated. The functionality previously controlled by KEYCARD is incorporated into the default processing of inline statistics during the execution of REBUILD INDEX STATISTICS and cannot be disabled.

Option descriptions

**INDEX creator-id.index-name**

Specifies an index that is to be reorganized.

creator-id. specifies the creator of the index and is optional. If you omit the qualifier creator ID, Db2 uses the user identifier for the utility job. index-name is the qualified name of the index to copy. For an index, you can specify either an index name or an index space name. Enclose the index name in quotation marks if the name contains a blank.

**INDEXSPACE database-name.index-space-name**

Specifies the qualified name of the index space that is obtained from the SYSIBM.SYSINDEXES table.

database-name specifies the name of the database that is associated with the index and is optional.

The default value is **DSNDB04**.

index-space-name specifies the qualified name of the index space that is to be reorganized; the name is obtained from the SYSIBM.SYSINDEXES table.

**LIST listdef-name**

Specifies the name of a previously defined LISTDEF list name. The INDEX keyword is required to differentiate this REORG INDEX LIST from REORG TABLESPACE LIST. The utility allows one LIST keyword for each control statement of REORG INDEX. The list must not contain any table spaces. REORG INDEX is invoked once for each item in the list. This utility will only process clone data if the CLONE keyword is specified. The use of CLONED YES on the LISTDEF statement is not sufficient.

Do not specify STATISTICS INDEX index-name with REORG INDEX LIST. If you want to collect inline statistics for a list of indexes, just specify STATISTICS.

You cannot specify DSNUM and PART with LIST on any utility.

The partitions or partition ranges can be specified in a list.
PART integer
Identifies a partition that is to be reorganized. You can reorganize a single
partition of a partitioning index. You cannot specify PART with LIST. integer is
the physical partition number. It must be in the range from 1 to the number of
partitions that are defined for the partitioning index. The maximum value is
4096.

integer designates a single partition.

If you omit the PART keyword, the entire index is reorganized.

REUSE
When used with SHRLEVEL NONE, specifies that REORG is to logically reset
and reuse DB2-managed data sets without deleting and redefining them. If you
do not specify REUSE and SHRLEVEL NONE, Db2 deletes and redefines
Db2-managed data sets to reset them.

If a data set has multiple extents and you use the REUSE parameter, the
extents are not released.

If you specify SHRLEVEL REFERENCE or CHANGE with REUSE, REUSE
does not apply

CLONE
Indicates that REORG INDEX is to reorganize only the specified index spaces
and indexes that are defined on clone tables. This utility will only process
clone data if the CLONE keyword is specified. The use of CLONED YES on
the LISTDEF statement is not sufficient.

SHRLEVEL
Specifies the method for performing the reorganization. The parameter
following SHRLEVEL indicates the type of access that is to be allowed during
the RELOAD phase of REORG.

NONE
Specifies that reorganization is to operate by unloading from the area that
is being reorganized (while applications can read but cannot write to the
area), building into that area (while applications have no access), and then
allowing read-write access again.

If you specify NONE (explicitly or by default), you cannot specify the
following parameters:
• MAXRO
• LONGLOG
• DELAY
• DEADLINE
• DRAIN_WAIT
• RETRY
• RETRY_DELAY

REFERENCE
Specifies that reorganization is to operate as follows:
• Unload from the area that is being reorganized while applications can
read but cannot write to the area.
• Build into a shadow copy of that area while applications can read but
cannot write to the original copy.
• Switch the future access of the applications from the original copy to the shadow copy by exchanging the names of the data sets, and then allowing read-write access again.

If you specify REFERENCE, you cannot specify the following parameters:
• UNLOAD (Reorganization with REFERENCE always performs UNLOAD CONTINUE.)
• MAXRO
• LONGLOG
• DELAY

CHANGE
Specifies that reorganization is to operate as follows:
• Unload from the area that is being reorganized while applications can read and write to the area.
• Build into a shadow copy of that area while applications can read and write to the original copy.
• Apply the log of the original copy to the shadow copy while applications can read and usually write to the original copy.
• Switch the future access of the applications from the original copy to the shadow copy by exchanging the names of the data sets, and then allowing read-write access again.

If you specify CHANGE, you cannot specify the UNLOAD parameter. Reorganization with CHANGE always performs UNLOAD CONTINUE.

SHRLEVEL CHANGE cannot be specified if the table space has the NOT LOGGED attribute.

DEADLINE
Specifies the deadline for the SWITCH phase to begin. If Db2 estimates that the SWITCH phase does not begin by the deadline, Db2 issues the messages that the DISPLAY UTILITY command issues and then terminates reorganization.

The final result and all the timestamp calculation of DEADLINE will be in TIMESTAMP(6).

NONE
Specifies that no deadline exists by which the switch phase of log processing must begin.

timestamp
Specifies the deadline for the switch phase of log processing to begin. This deadline must not have already occurred when REORG is executed.

labeled-duration-expression
Calculates the deadline for the switch phase of log processing to begin. The calculation is based on either CURRENT_TIMESTAMP or CURRENT_DATE. You can add or subtract one or more constant values to specify the deadline. This deadline must not have already occurred when REORG is executed. CURRENT_TIMESTAMP and CURRENT_DATE are evaluated once, when the REORG statement is first processed. If a list of objects is specified, the same value will be in effect for all objects in the list.

CURRENT_DATE
Specifies that the deadline is to be calculated based on the CURRENT DATE.
CURRENT_TIMESTAMP

Specifies that the deadline is to be calculated based on the CURRENT TIMESTAMP.

WITH TIME ZONE

Specifies that the CURRENT_TIMESTAMP is compared with the time zone column. The timestamp precision of the special register CURRENT_TIMESTAMP should be the same as the column timestamp precision. Otherwise the default timestamp precision is used. The time zone of CURRENT_TIMESTAMP is the value of special register CURRENT_TIMEZONE. The comparison is done by comparing the UTC portion of the timestamp.

constant

Indicates a unit of time and is followed by one of the seven duration keywords: YEARS, MONTHS, DAYS, HOURS, MINUTES, SECONDS, or MICROSECONDS. The singular form of these words is also acceptable: YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, MICROSECOND.

If REORG SHRLEVEL REFERENCE or SHRLEVEL CHANGE terminates because of a DEADLINE specification, Db2 issues message DSNU374I with reason code 2 but does not set a restrictive status.

DRAIN_WAIT integer

Specifies the number of seconds that the utility waits when draining for SQL statements (inserts, updates, deletes, and selects). The specified time is the aggregate time for all partitions of the index that is to be reorganized. This value overrides the values specified by IRLMRWT and UTIMOUT, for these SQL statements only. For operations like commands, the IRLMRWT and UTIMOUT values are used. Valid values for integer are from 0 to 1800. If the keyword is omitted or if a value of 0 is specified, the utility uses the value of the lock timeout system parameter IRLMRWT.

RETRY integer

Specifies the maximum number of retries that REORG is to attempt. Valid values for integer are from 0 to 255.

Specifying RETRY can lead to increased processing costs and can result in multiple or extended periods of read-only access.

The default value is the value of the UTIMOUT subsystem parameter.

RETRY_DELAY integer

Specifies the minimum duration, in seconds, between retries. Valid values for integer are from 1 to 1800.

If you do not specify RETRY_DELAY, REORG INDEX uses the smaller of the following two values:

- DRAIN_WAIT value × RETRY value
- DRAIN_WAIT value × 10

MAXRO integer

Specifies the maximum amount of time for the last iteration of log processing. During that iteration, applications have read-only access.

The actual execution time of the last iteration might exceed the specified MAXRO value.

The ALTER UTILITY command can change the value of MAXRO.
The default value is the **RETRY_DELAY** default value.

**integer**

integer is the number of seconds. Specifying a small positive value reduces the length of the period of read-only access, but it might increase the elapsed time for REORG to complete. If you specify a huge positive value, the second iteration of log processing is probably the last iteration.

The default value is **300** seconds.

**DEFER**

Specifies that the iterations of log processing with read-write access can continue indefinitely. REORG never begins the final iteration with read-only access, unless you change the MAXRO value by using the **ALTER UTILITY** command.

If you specify DEFER, you should also specify LONGLOG CONTINUE.

If you specify DEFER, and Db2 determines that the actual time for an iteration and the estimated time for the next iteration are both less than 5 seconds, Db2 adds a 5-second pause to the next iteration. This pause reduces consumption of processor time. The first time this situation occurs for a given execution of REORG, Db2 sends message DSNU362I to the console. The message states that the number of log records that must be processed is small and that the pause occurs. To change the MAXRO value and thus cause REORG to finish, execute the **ALTER UTILITY** command. Db2 adds the pause whenever the situation occurs; however, Db2 sends the message only if 30 minutes have elapsed since the last message was sent for a given execution of REORG.

**DRAIN**

Specifies drain behavior at the end of the log phase after the MAXRO threshold is reached and when the last iteration of the log is to be applied.

**WRITERS**

Specifies that Db2 drains only the writers during the log phase after the MAXRO threshold is reached and then issues DRAIN ALL on entering the switch phase.

**ALL**

Specifies the current default action, in which Db2 is to drain all readers and writers during the log phase, after the MAXRO threshold is reached.

Consider specifying DRAIN ALL if the following conditions are both true:

- SQL update activity is high during the log phase.
- The default behavior results in a large number of -911 SQL error messages.

**LONGLOG**

Specifies the action that Db2 is to perform, after sending a message to the console, if the number of records that the next iteration of log process is to process is not sufficiently lower than the number that the previous iterations processed. This situation means that REORG INDEX is not reading the application log quickly enough to keep pace with the writing of the application log.

**CONTINUE**

Specifies that until the time on the JOB statement expires, Db2 is to continue performing reorganization, including iterations of log processing, if the estimated time to perform an iteration exceeds the time that is specified with MAXRO.
A value of DEFER for MAXRO and a value of CONTINUE for LONGLOG together mean that REORG INDEX is to continue allowing access to the original copy of the area that is being reorganized and does not switch to the shadow copy. The user can execute the ALTER UTILITY command with a large value for MAXRO when the switching is wanted.

**TERM**

Specifies that Db2 is to terminate reorganization after the delay specified by the DELAY parameter.

**DRAIN**

Specifies that Db2 is to drain the write claim class after the delay that is specified by the DELAY parameter. This action forces the final iteration of log processing to occur.

**DELAY integer**

Specifies the minimum interval between the time that REORG sends the LONGLOG message to the console and the time REORG that performs the action that is specified by the LONGLOG parameter.

*integer* is the number of seconds.

The default value is **1200**.

**TIMEOUT**

Specifies the action that is to be taken if the REORG INDEX utility gets a timeout condition while trying to drain objects in either the log or switch phases.

**TERM**

Indicates that Db2 is to behave as follows if you specify the TERM option and a time out condition occurs:

1. Db2 issues an implicit TERM UTILITY command, causing the utility to end with a return code 8.
2. Db2 issues the DSNU590I and DSNU170I messages.
3. Db2 leaves the objects in a RW state.

**ABEND**

Indicates that if a timeout condition occurs, Db2 is to leave the objects in a UTRO or UTUT state.

**LOGRANGES**

Specifies whether REORG is to use SYSLGRNX information for the LOG phase.

**YES**

REORG uses SYSLGRNX information for the LOG phase whenever possible. This option is the default behavior.

**NO**

REORG does not use SYSLGRNX information for the LOG phase. This option can cause REORG to run much longer. In a data sharing environment this option can result in the merging of all logs from all members. This option is feasible when there is a known integrity issue with SYSLGRNX entries and performance problems in accessing SYSLGRNX for log read determination.

**SWITCHTIME**

Specifies the time for the final log iteration of the LOG phase to begin. The final result and all of the time stamp calculations of SWITCHTIME are in TIMESTAMP(6). This keyword can be specified with the MAXRO keyword. If MAXRO DEFER is not specified, REORG enters the final log iteration of the LOG phase before the specified SWITCHTIME value if the specified or
defaulted MAXRO criteria is met. When MAXRO DEFER is specified, REORG does not attempt to enter to the final log iteration until the specified SWITCHTIME is met or affected by an external ALTER UTILITY command in the changing of its MAXRO value.

**NONE**
Does not specify a time for the final log iteration of the LOG phase. This option is the default behavior.

**timestamp**
Specifies the time that the final log iteration of the LOG phase is to begin. This time must not have already occurred when REORG is run.

**labeled-duration-expression**
Calculates the time for the final log iteration of LOG phase is to begin. The calculation is based on either CURRENT TIMESTAMP or CURRENT DATE. You can add or subtract one or more constant values to specify the switch time. This switch time must not have already occurred when REORG is run. CURRENT TIMESTAMP and CURRENT DATE are evaluated once, when the REORG statement is first processed. If a list of objects is specified, the same value is in effect for all objects in the list.

**CURRENT_DATE**
Specifies that the deadline is to be calculated based on the CURRENT DATE.

**CURRENT_TIMESTAMP**
Specifies that the deadline is to be calculated based on the CURRENT TIMESTAMP.

**WITH TIME ZONE**
Specifies that the CURRENT TIMESTAMP is compared with the time zone column. The time stamp precision of the special register CURRENT_TIMESTAMP should be the same as the column time stamp precision. Otherwise, the default time stamp precision is used. The time zone of CURRENT_TIMESTAMP is the value of special register CURRENT TIMEZONE. The comparison is done by comparing the Coordinated Universal Time portion of the time stamp.

**constant**
Indicates a unit of time and is followed by one of the seven duration keywords: YEARS, MONTHS, DAYS, HOURS, MINUTES, SECONDS, or MICROSECONDS. The singular form of these words is also acceptable: YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, MICROSECOND.

**NEWMAXRO**
Specifies the maximum amount of time for the last log iteration after SWITCHTIME is met. The SWITCHTIME keyword must also be specified. This value overrides the existing MAXRO parameter that is specified. The default is NONE.

**NONE**
Specifies that when the specified SWITCHTIME is met, REORG proceeds to the last log iteration without taking log processing time into consideration. Specifying NONE results in REORG entering the last log iteration almost immediately at or after the specified SWITCHTIME. This option is the default.

**integer**
integer is the number of seconds. Specifying a small positive value reduces
the length of the period of read-only access, but it might increase the elapsed time for REORG to complete. Specifying a large positive value probably ensures that REORG will enter the last log iteration almost immediately at or after the specified SWITCHTIME.

**FORCE**

Specifies the action to be taken when the utility is draining the table space.

When REORG FORCE is canceling the threads, it performs a soft cancel similar to the cancel that the CANCEL THREAD does.

**NONE**

Specifies that no action is taken when REORG performs drain. The REORG utility waits for the claimers to commit. The utility will timeout or restart when the drain fails, as determined by existing conditions.

**READERS**

Specifies that read claimers are canceled when REORG is requesting a drain all on the last RETRY processing.

**ALL**

specifies that both read and write claimers are canceled when REORG is requesting a drain all or drain writers on the last RETRY processing.

**FASTSWITCH**

Specifies which switch methodology is to be used for a reorganization.

When FASTSWITCH is specified with SHRLEVEL CHANGE or SHRLEVEL REFERENCE, the UTILITY_OBJECT_CONVERSION subsystem parameter setting NONE, BASIC, or EXTENDED is accepted.

**YES**

Enables the SWITCH phase to use the FASTSWITCH methodology. This option is not allowed for the catalog (DSNDB06) or directory (DSNDB01).

**NO**

Causes the SWITCH phase to use IDCAMS RENAME.

**LEAFDISTLIMIT** integer

The LEAFDISTLIMIT option is deprecated, and the alternative is running DSNACCOX. Specifies that the value for integer is to be compared to the LEAFDIST value for the specified partitions of the specified index in SYSIBM.SYSINDEXPART. If any LEAFDIST value exceeds the specified LEAFDISTLIMIT value, REORG is performed or, if you specify REPORTONLY, recommended.

The default value is 200.

Because a node ID index, auxiliary index, hash index, or XML index has a LEAFDIST value of -2, REORG is not performed for any of those indexes when LEAFDISTLIMIT is specified.

**REPORTONLY**

The REPORTONLY option is deprecated, and the alternative is running DSNACCOX. Specifies that REORG is only to be recommended, not performed. REORG produces a report with one of the following return codes:

1 No limit met; no REORG performed or recommended.

2 REORG performed or recommended.

**UNLOAD**

Specifies whether the utility job is to continue processing or terminate after the data is unloaded.
CONTINUE

Specifies that, after the data has been unloaded, the utility is to continue processing.

PAUSE

The UNLOAD PAUSE option has been deprecated. If you need to stop the utility after the keys are unloaded, use DIAGNOSE in combination with the REORG utility to stop the process (pause).

Specifies that, after the data has been unloaded, processing is to end. The utility stops and the RELOAD status is stored in SYSIBM.SYSUTIL so that processing can be restarted with RELOAD RESTART(PHASE).

This option is useful if you want to redefine data sets during reorganization. For example, with a user-defined data set, you can:

• Run REORG with the UNLOAD PAUSE option.
• Redefine the data set using Access Method Services.
• Restart REORG by resubmitting the previous job and specifying RESTART(PHASE).

If no records are unloaded during an UNLOAD PAUSE, when REORG is restarted, the RELOAD and BUILD phases are bypassed.

You cannot use UNLOAD PAUSE if you specify the LIST option.

ONLY

The UNLOAD ONLY option has been deprecated. If you need to unload the keys, use DIAGNOSE in combination with the REORG utility to stop the process after the keys are unloaded, and TERM UTIL to terminate the utility.

Specifies that, after the data has been unloaded, the utility job ends and the status in SYSIBM.SYSUTIL that corresponds to this utility ID is removed.

STATISTICS

Specifies that statistics for the index are to be collected; the statistics are either reported or stored in the Db2 catalog. You cannot collect inline statistics for indexes on the catalog and directory tables.

Restrictions:

• If you specify STATISTICS for encrypted data, Db2 might not provide useful information on this data.
• You cannot specify STATISTICS for clone objects.

If pending definition changes are materialized during REORG INDEX with SHRLEVEL REFERENCE or SHRLEVEL CHANGE, index statistics are collected and updated in the Db2 catalog by default.

If the STATISTICS keyword was not specified in the REORG INDEX with SHRLEVEL REFERENCE or CHANGE statement when pending definition changes are materialized, the following keywords are run by default: STATISTICS UPDATE ALL HISTORY ALL. If you also specify the STATISTICS keyword in the REORG INDEX with the SHRLEVEL REFERENCE or SHRLEVEL CHANGE statement when pending definition changes are materialized, the options specified overwrite the default options.

Recommendation: Some partition statistics can become obsolete due to the materialization of pending definition changes. The partition statistics that can become obsolete are COLGROUP statistics, statistics for key column values in
indexes, HISTOGRAM statistics, frequency statistics with NUMCOLS > 1, and statistics for extended indexes where applicable. Run the RUNSTATS utility to collect the partition statistics again.

**REPORT**

Indicates whether a set of messages to report the collected statistics is to be generated.

**NO**

Indicates that the set of messages is not to be sent as output to SYSPRINT.

**YES**

Indicates that the set of messages is to be sent as output to SYSPRINT. The generated messages are dependent on the combination of keywords (such as TABLESPACE, INDEX, TABLE, and COLUMN) that you specify with the RUNSTATS utility. However, these messages are not dependent on the specification of the UPDATE option. REPORT YES always generates a report of SPACE and ACCESSPATH statistics.

**KEYCARD**

The KEYCARD option is deprecated in the utility control statement and no longer needs to be specified to collect cardinality statistics on the values in the key columns of an index.

When the STATISTICS option is specified, the utility automatically collects all of the distinct values in all of the 1 to \( n \) key column combinations for the indexes being rebuilt. \( n \) is the number of columns in the index. With the deprecation of KEYCARD, this functionality cannot be disabled.

The utility tolerates the specification of the KEYCARD option. The utility does not issue any messages if the control statement includes or excludes the KEYCARD option when STATISTICS is specified.

**FREQVAL**

Controls the collection of frequent-value statistics. If you specify FREQVAL, it must be followed by the NUMCOLS keyword.

**NUMCOLS**

Indicates the number of key columns that are to be concatenated when collecting frequent values from the specified index. If you specify 3, the utility collects frequent values on the concatenation of the first three key columns.

The default value is 1, which means that Db2 is to collect frequent values only on the first key column of the index.

**COUNT**

Indicates the number of frequent values that are to be collected. If you specify 15, the utility collects 15 frequent values from the specified key columns.

The default value is 10.

**HISTOGRAM**

Indicates that histogram statistics are requested for the specified index.

**NUMCOLS**

The number of key columns that are to be concatenated when collecting histogram statistics from the specified index.

**NUMQUANTILES**

The integer values that follows NUMQUANTILES indicates the number of quantiles are requested. The integer value must be greater than or equal to 1.
Histogram statistics can be collected only on keys with the same order. If the specified key columns for histogram statistics are of mixed or random order, a DSNU633I warning message is issued.

Histogram statistics that you collect through REBUILD INDEX are not the same as histogram statistics that you collect through RUNSTATS. Histogram statistics that you collect with REBUILD INDEX are only rough estimates. To obtain more exact statistics, use RUNSTATS.

**Related information:**

**UPDATE**

Indicates whether the collected statistics are to be inserted into the catalog tables. UPDATE also allows you to select statistics that are used for access path selection or statistics that are used by database administrators.

- **ALL**
  Indicates that all collected statistics are to be updated in the catalog.

- **ACCESSPATH**
  Indicates that the only catalog table columns that are to be updated are those that provide statistics that are used for access path selection.

- **SPACE**
  Indicates that the only catalog table columns that are to be updated are those that provide statistics to help the database administrator assess the status of a particular table space or index.

- **NONE**
  Indicates that catalog tables are not to be updated with the collected statistics. This option is valid only when REPORT YES is specified.

**HISTORY**

Records all catalog table inserts or updates to the catalog history tables.

The default is supplied by the value that is specified in STATISTICS HISTORY on panel DSNTIP6.

- **ALL**
  Indicates that all collected statistics are to be updated in the catalog history tables.

- **ACCESSPATH**
  Indicates that the only catalog history table columns that are to be updated are those that provide statistics that are used for access path selection.

- **SPACE**
  Indicates that only space-related catalog statistics are to be updated in catalog history tables.

- **NONE**
  Indicates that catalog history tables are not to be updated with the collected statistics.

**FORCEROLLUP**

Specifies whether aggregation or rollup of statistics is to take place when you execute RUNSTATS even if some indexes or index partitions are empty. This keyword enables the optimizer to select the best access path.

The following options are available for the **FORCEROLLUP** keyword:
YES
Indicates that forced aggregation or rollup processing is to be done, even though some indexes or index partitions might not contain data.

NO
Indicates that aggregation or rollup is to be done only if data is available for all indexes or index partitions.

If data is not available, the utility issues DSNU623I message if you have set the installation value for STATISTICS ROLLUP on panel DSNTIP6 to NO.

SORTDEVT device-type
Specifies the device type for temporary data sets that are to be dynamically allocated by the external sort program. For device-type, specify any disk device that is valid on the DYNALLOC parameter of the SORT or OPTION options for the sort program. Tape devices are not supported by the sort program.

SORTNUM integer
Specifies the number of temporary data sets that are to be dynamically allocated when collecting statistics for a data-partitioned secondary index. If you omit SORTDEVT, SORTNUM is ignored. If you use SORTDEVT and omit SORTNUM, no value is passed to the sort program; the sort program uses its own default.

integer is the number of temporary data sets that can range from 2 to 255.

REORG INDEX does not sort index keys. Only one sort can be performed, and that is if inline statistics are being collected for a DPSI.

Important: The SORTNUM keyword is ignored if the UTSORTAL subsystem parameter is set to YES and the IGN SORTN subsystem parameter is set to YES.

WORKDDN(ddname)

ddname specifies the DD statement for the unload data set.

ddname
Is the DD name of the temporary work file for build input.

The default value is SYSUT1.

The WORKDDN keyword specifies either a DD name or a TEMPLATE name from a previous TEMPLATE control statement. If utility processing detects that the specified name is both a DD name in the current job step and a TEMPLATE name, the utility uses DD name.

Even though WORKDDN is an optional keyword, a DD statement for the unload output data set is required in the JCL. If you do not specify WORKDDN, or if you specify it without ddname, the JCL must have a DD statement with the name SYSUT1. If ddname is given, you must provide a DD statement or TEMPLATE that matches the DD name.

PREFORMAT
Specifies that the remaining pages are to be preformatted up to the high-allocated RBA in the index space. The preformatting occurs after the index is built.

PREFORMAT can operate on an entire index space, or on a partition of a partitioned index space.

PREFORMAT is ignored if you specify UNLOAD ONLY.

FLASHCOPY
Specifies whether FlashCopy technology is used to create a copy of the object.
Valid values are YES, NO, or CONSISTENT. When FlashCopy is used, a separate data set is created for each partition or piece of the object.

The FlashCopy specifications on the utility control statement override any specifications for FlashCopy that are defined by using the Db2 subsystem parameters. If the FlashCopy subsystem parameters specify the use of FlashCopy as the default behavior of this utility, the FLASHCOPY option can be omitted from the utility control statement.

**Important:** If the input data set is less than one cylinder, FlashCopy technology might not be used for copying the objects regardless of the FLASHCOPY settings. The copy is performed by IDCAMS if FlashCopy is not used.

**NO**  Specifies that no FlashCopy is made. NO is the default value for FLASHCOPY.

**YES**  Specifies that FlashCopy technology is used to copy the object.

Specify YES only if the Db2 data sets are on FlashCopy Version 2 disk volumes.

**Important:** Under the following circumstances, the REORG INDEX utility might not use FlashCopy even though YES is specified:
- FlashCopy Version 2 disk volumes are not available
- The source tracks are already the target of a FlashCopy operation
- The target tracks are the source of a FlashCopy operation
- The maximum number of relationships for the copy is exceeded

In the event that FlashCopy is not used, the REORG INDEX utility uses traditional I/O methods to copy the object, which can result in longer than expected execution time.

If SHRLEVEL REFERENCE or SHRLEVEL CHANGE is specified when the copy operation is forced to use traditional I/O methods, an even longer outage might occur, because the FlashCopy image copies are created during the SWITCH phase of utility execution.

**Related information:**
- “FlashCopy image copies” on page 157
- “Subsystem parameters for refining DFSMSdss COPY operation with utilities” on page 40

**CONSISTENT**  Specifies that FlashCopy technology is used to copy the object. Because the copies created by the REORG INDEX utility are already consistent, the utility treats a specification of CONSISTENT the same as a specification of YES.

**RBALRSN_CONVERSION**  Specifies the RBA or LRSN format of the target object after the completion of the REORG utility. If the keyword is not specified, the conversion specified in the UTILITY_OBJECT_CONVERSION subsystem parameter is accepted.

**NONE**  Specifies that no conversion is performed.
The utility fails if RBALRSN_CONVERSION NONE is specified on a table space that is in basic 6-byte format and the
UTILITY_OBJECT_CONVERSION subsystem parameter is set to NOBASIC.

**BASIC**
Specifies that if an object is found in extended 10-byte format, it is converted to 6-byte basic format.
The utility fails if RBALRSN_CONVERSION BASIC is specified and the
UTILITY_OBJECT_CONVERSION subsystem parameter is set to NOBASIC.

**EXTENDED**
Specifies that if an object is found in basic 6-byte format, it is converted to 10-byte extended format.

If RBALRSN_CONVERSION is not specified, and subsystem parameter
UTILITY_OBJECT_CONVERSION is set to BASIC or EXTENDED,
RBALRSN_CONVERSION defaults to the UTILITY_OBJECT_CONVERSION value. If RBALRSN_CONVERSION is not specified, and subsystem parameter
UTILITY_OBJECT_CONVERSION is set to NOBASIC,
RBALRSN_CONVERSION defaults to EXTENDED.

If a CLONE relationship exists, the page set conversion cannot be performed. For clone relationships, you must drop the clone table, convert the base table to extended 10-byte format, and then re-create the clone table.

**FCCOPYDDN**
Specifies the template to be used to create the FlashCopy image copy data set names. If a value is not specified for FCCOPYDDN on the REORG INDEX control statement when FlashCopy is used, the value specified on the FCCOPYDDN subsystem parameter determines the template to be used.

(template-name)
The data set names for the FlashCopy image copy are allocated according to the template specification. For table space or index space level FlashCopy image copies, because a data set is allocated for each partition or piece, ensure that the data set naming convention in the template specification is unique enough. Use the &DSNUM variable, which resolves to a partition number or piece number at execution time.

Related tasks:
"Improving LOAD performance" on page 336

Related reference:
Chapter 34, “TEMPLATE,” on page 793
Chapter 18, “LISTDEF,” on page 215

Related information:
Db2 Sort for z/OS

Before running REORG INDEX
Certain activities might be required before you run the REORG INDEX utility, depending on your situation.
Region size

The recommended minimum region size is 4096 KB.

Restart-pending status and SHRLEVEL CHANGE

If you specify SHRLEVEL CHANGE, REORG drains the write claim class near the end of REORG processing. In a data sharing environment, if a data sharing member fails and that member has restart-pending status for a target page set, the drain can fail. You must postpone running REORG with SHRLEVEL CHANGE until all restart-pending statuses have been removed. You can use the DISPLAY GROUP command to determine whether a member's status is FAILED. You can use the DISPLAY DATABASE command with the LOCKS option to determine if locks are held.

Data sharing considerations for REORG

You must not execute REORG on an object if another Db2 subsystem holds retained locks on the object or has long-running noncommitting applications that use the object. You can use the DISPLAY GROUP command to determine whether a member's status is "FAILED." You can use the DISPLAY DATABASE command with the LOCKS option to determine if locks are held.

Fallback recovery considerations

Successful REORG INDEX processing inserts a SYSCOPY row with ICTYPE='W' for an index that was defined with COPY YES. REORG also places a reorganized index in informational COPY-pending status. You should take a full image copy of the index after the REORG job completes to create a valid point of recovery.

RECOVER-pending and REBUILD-pending status

You cannot reorganize an index if any partition of the index is in the RECOVER-pending status or in the REBUILD-pending status. Similarly, you cannot reorganize a single index partition if it is in the RECOVER-pending status or in the REBUILD-pending status.

The RECOVER-pending restrictive state is:

RECP The index space or partition is in a RECOVER-pending status. A single logical partition in RECP does not restrict access to other logical partitions that are not in RECP. You can reset RECP by recovering only the single logical partition.

The REBUILD-pending restrictive states are:

RBDP REBUILD-pending status is set on a physical or logical index partition. The individual physical or logical partition is inaccessible; you must rebuild the object using the REBUILD INDEX utility.

PSRBD Page set REBUILD-pending (PSRBD) is set for nonpartitioning indexes. The entire index space is inaccessible; you must rebuild the object by using the REBUILD INDEX utility.

RBDP* A REBUILD-pending status is set only on logical partitions of
nonpartitioning indexes. The entire index is inaccessible, but it is made available again when you rebuild the affected partitions by using the REBUILD INDEX utility.

CHECK-pending status

You cannot reorganize an index when the data is in the CHECK-pending status.

Running REORG INDEX when the index has a VARBINARY column

If you run REORG INDEX against an index with the following characteristics, REORG INDEX fails:

- The index was created on a VARBINARY column or a column with a distinct type that is based on a VARBINARY data type.
- The index column has the DESC attribute.

To fix the problem, alter the column data type to BINARY, and then rebuild the index.

Related reference:

“RECOVER-pending status” on page 1044
“REBUILD-pending status” on page 1043
Chapter 11, “CHECK DATA,” on page 73

Data sets that REORG INDEX uses

The REORG INDEX utility uses a number of data sets during its operation.

The following table lists the data sets that REORG uses. The table lists the DD name that is used to identify the data set, a description of the data set, and an indication of whether it is required. Include statements in your JCL for each required data set and any optional data sets that you want to use.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYsin</td>
<td>Input data set that contain the utility control statement.</td>
<td>Yes</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>Output data set for messages.</td>
<td>Yes</td>
</tr>
<tr>
<td>STPRIN01</td>
<td>A data set that contains messages from the sort program (usually, SYSOUT or DUMMY). This data set is used when statistics are collected on at least one data-partitioned secondary index.</td>
<td>No</td>
</tr>
<tr>
<td>Work data set</td>
<td>A temporary data set for unload output and build input. Specify the DD or template name with the WORKDDN option of the utility control statement. The default DD name is SYSUT1.</td>
<td>Yes</td>
</tr>
<tr>
<td>Sort work data sets</td>
<td>Temporary data sets for sort input and output when collecting inline statistics on at least one data-partitioned secondary index. The DD names have the form ST01WKnn.</td>
<td>No</td>
</tr>
</tbody>
</table>
Table 69. Data sets that REORG INDEX uses (continued)

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTPRINT</td>
<td>A data set that contains messages from the sort program (usually, SYSOUT or DUMMY).</td>
<td>Yes</td>
</tr>
<tr>
<td>FlashCopy image copy data sets</td>
<td>For copies of the entire index space, a separate VSAM data set for each partition or piece that is contained in the index space. For partition-level or piece-level copies, a VSAM data set for each partition or piece that is being copied.</td>
<td>No</td>
</tr>
</tbody>
</table>

Note:
1. STPRIN01 is required if statistics are being collected on at least one data-partitioned secondary index, but REORG INDEX dynamically allocates the STPRIN01 data set if UTPRINT is allocated to SYSOUT.
2. Required when collecting inline statistics on at least one data-partitioned secondary index.
3. If the DYNALLOC parm of the SORT program is not turned on, you need to allocate the data set. Otherwise, the sort program dynamically allocates the temporary data set.
4. It is recommended that you use dynamic allocation by specifying SORTDEVT in the utility statement because dynamic allocation reduces the maintenance required of the utility job JCL.
5. Required if you specify the FLASHCOPY YES

The following objects are named in the utility control statement and do not require DD statements in the JCL:

Index  Object to be reorganized.

Calculating the size of the work data sets

When reorganizing an index space, you need a non-DB2 sequential work data set. That data set is identified by the DD statement that is named in the WORKDDN option. During the UNLOAD phase, the index keys and the data pointers are unloaded to the work data set. This data set is used to build the index. It is required only during the execution of REORG.

Use the following formula to calculate the approximate size (in bytes) of the WORKDDN data set SYSUT1:

\[
\text{size} = \text{number of keys} \times (\text{key length} + 8)
\]

Calculating the size of the sort work data sets

To calculate the approximate size (in bytes) of the ST01WK\(n\) data set, use the following formula:

\[
2 \times (\text{maximum record length} \times \text{numcols} \times (\text{count} + 2) \times \text{number of indexes})
\]

The variables in the preceding formula have the following values:

maximum record length

Maximum record length of the SYSCOLDISTSTATS record that is processed when collecting frequency statistics (You can obtain this value from the RECLENGTH column in SYSTABLES.)
**numcols**
Number of key columns to concatenate when you collect frequent values from the specified index.

**count** Number of frequent values that Db2 is to collect.

Sort work data sets cannot span volumes. Smaller volumes require more sort work data sets to sort the same amount of data; therefore, large volume sizes can reduce the number of needed sort work data sets. When you allocate sort work data sets on disk, the recommended amount of space to allow provides at least 1.2 times the amount of data that is to be sorted.

Tape devices are not supported for sort work data sets.

### Changing data set definitions

If the index space is defined by storage groups, space allocation is handled by Db2 and data set definitions cannot be altered during the reorganization process. Db2 deletes and redefines the necessary data sets to reorganize the object.

For REORG with SHRLEVEL REFERENCE or CHANGE, you can use the ALTER STOGROUP command to change the characteristics of a Db2-managed data set. You can effectively change the characteristics of a user-managed data set by specifying the new characteristics when creating the shadow data set. In particular, placing the original and shadow data sets on different disk volumes might reduce contention and thus improve the performance of REORG and the performance of applications during REORG execution.

**Related concepts:**
"Data sets that online utilities use" on page 16

**Related reference:**
Db2 Sort for z/OS

**Related information:**
DFSORT Application Programming Guide

### Shadow data sets for REORG INDEX

When you execute the REORG INDEX utility with SHRLEVEL REFERENCE or SHRLEVEL CHANGE, the utility uses shadow data sets.

For user-managed data sets, you must preallocate the shadow data sets before you execute REORG INDEX with SHRLEVEL REFERENCE or SHRLEVEL CHANGE. If an index or partitioned index resides in Db2-managed data sets and shadow data sets do not already exist when you execute REORG INDEX, Db2 creates the shadow data sets. At the end of REORG processing, the Db2-managed shadow data sets are deleted. You can create the shadows ahead of time for Db2-managed data sets.

**Shadow data set names**

Each shadow data set must have the following name:

`catname.DSNDBx.dbname.psname.y000z.Lnnn`

In the preceding name, the variables have the following meanings:
variable

meaning

catname
The VSAM catalog name or alias

x
C or D

dbname
Database name

psname
Table space name or index name

y
1 or J

z
1 or 2

Lnnn
Partition identifier. Use one of the following values:
- A001 through A999 for partitions 1 through 999
- B000 through B999 for partitions 1000 through 1999
- C000 through C999 for partitions 2000 through 2999
- D000 through D999 for partitions 3000 through 3999
- E000 through E996 for partitions 4000 through 4096

GUP! GUP!

To determine the names of existing data sets, execute one of the following queries against the SYSTABLEPART or SYSINDEXPART catalog tables:

```
SELECT DBNAME, TSNAME, IPREFIX
FROM SYSIBM.SYSTABLEPART
WHERE DBNAME = 'dbname'
  AND TSNAME = 'psname';

SELECT DBNAME, IXNAME, IPREFIX
FROM SYSIBM.SYSINDEXES X, SYSIBM.SYSINDEXPART Y
WHERE X.NAME = Y.IXNAME
  AND X.CREATOR = Y.IXCREATOR
  AND X.DBNAME = 'dbname'
  AND X.INDEXSPACE = 'psname';
```

Defining shadow data sets

Consider the following actions when you preallocate the data sets:
- Allocate the shadow data sets according to the rules for user-managed data sets.
- Define the shadow data sets as LINEAR.
- Use SHAREOPTIONS(3,3).
- Define the shadow data sets as EA-enabled if the original table space or index space is EA-enabled.
- Allocate the shadow data sets on the volumes that are defined in the storage group for the original table space or index space.

If you specify a secondary space quantity, Db2 does not use it. Instead, Db2 uses the SECQTY value for the table space or index space.
**Recommendation:** Use the MODEL option, which causes the new shadow data set to be created like the original data set. This method is shown in the following example:

```sql
DEFINE CLUSTER +
   (NAME('catname.DSNDBC.dbname.psname.x0001.L001') +
    MODEL('catname.DSNDBC.dbname.psname.y0001.L001')) +
DATA +
   (NAME('catname.DSNDBD.dbname.psname.x0001.L001') +
    MODEL('catname.DSNDBD.dbname.psname.y0001.L001'))
```

**Creating shadow data sets for indexes**

Db2 treats preallocated shadow data sets as Db2-managed data sets.

When you preallocate shadow data sets for indexes, create the data sets as follows:

- Create shadow data sets for the partition of the table space and the corresponding partition in each partitioning index and data-partitioned secondary index.
- Create a shadow data set for logical partitions of nonpartitioned secondary indexes.

Use the same naming scheme for these index data sets as you use for other data sets that are associated with the base index, except use J0001 instead of I0001.

**Estimating the size of shadow data sets**

If you do not change the value of FREEPAGE or PCTFREE, the amount of space that is required for a shadow data set is approximately comparable to the amount of space that is required for the original data set.

**Concurrency and compatibility for REORG INDEX**

The REORG INDEX utility has certain concurrency and compatibility characteristics associated with it.

Db2 treats individual index partitions as distinct target objects. Utilities that operate on different partitions of the same index space are compatible.

**Claims**

The following table shows which claim classes REORG INDEX drains and any restrictive state the utility sets on the target object. The target is an index or index partition.

<table>
<thead>
<tr>
<th>Phase</th>
<th>REORG INDEX SHRLEVEL NONE</th>
<th>REORG INDEX SHRLEVEL REFERENCE</th>
<th>REORG INDEX SHRLEVEL CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNLOAD</td>
<td>DW/UTRO</td>
<td>DW/UTRO</td>
<td>CR/UTRW</td>
</tr>
<tr>
<td>BUILD</td>
<td>DA/UTUT</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>Last iteration of LOG</td>
<td>n/a</td>
<td>DA/UTUT(^1)</td>
<td>DW/UTRO</td>
</tr>
<tr>
<td>SWITCH</td>
<td>n/a</td>
<td>DA/UTUT</td>
<td>DA/UTUT</td>
</tr>
</tbody>
</table>
Table 70. Claim classes of REORG INDEX operations (continued)

<table>
<thead>
<tr>
<th>Phase</th>
<th>REORG INDEX SHRLEVEL NONE</th>
<th>REORG INDEX SHRLEVEL REFERENCE</th>
<th>REORG INDEX SHRLEVEL CHANGE</th>
</tr>
</thead>
</table>

Legend:
- CR: Claim the read claim class.
- DA: Drain all claim classes, no concurrent SQL access.
- DR: Drain the repeatable read class, no concurrent access for SQL repeatable readers.
- DW: Drain the write claim class, concurrent access for SQL readers.
- UTRO: Utility restrictive state, read only access allowed.
- UTUT: Utility restrictive state, exclusive control.
- none: Any claim, drain, or restrictive state for this object does not change in this phase.

Note:
1. Applicable if you specified DRAIN ALL.

Compatibility

The following table shows which utilities can run concurrently with REORG INDEX on the same target object. The target object can be an index space or a partition. If compatibility depends on particular options of a utility, that is also shown. REORG INDEX does not set a utility restrictive state if the target object is an index on DSND01.SYSUTILX.

Table 71. Compatibility of REORG INDEX with other utilities

<table>
<thead>
<tr>
<th>Action</th>
<th>REORG INDEX SHRLEVEL NONE, REFERENCE, or CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHECK DATA</td>
<td>No</td>
</tr>
<tr>
<td>CHECK INDEX</td>
<td>No</td>
</tr>
<tr>
<td>CHECK LOB</td>
<td>Yes</td>
</tr>
<tr>
<td>COPY INDEXSPACE</td>
<td>No</td>
</tr>
<tr>
<td>COPY TABLESPACE</td>
<td>Yes</td>
</tr>
<tr>
<td>DIAGNOSE</td>
<td>Yes</td>
</tr>
<tr>
<td>LOAD</td>
<td>No</td>
</tr>
<tr>
<td>MERGECOPY</td>
<td>Yes</td>
</tr>
<tr>
<td>MODIFY</td>
<td>Yes</td>
</tr>
<tr>
<td>QUIESCE</td>
<td>No</td>
</tr>
<tr>
<td>REBUILD INDEX</td>
<td>No</td>
</tr>
<tr>
<td>RECOVER INDEX</td>
<td>No</td>
</tr>
<tr>
<td>RECOVER INDEXSPACE</td>
<td>No</td>
</tr>
<tr>
<td>RECOVER TABLESPACE (with no options)</td>
<td>Yes</td>
</tr>
<tr>
<td>RECOVER TABLESPACE ERROR RANGE</td>
<td>Yes</td>
</tr>
<tr>
<td>RECOVER TABLESPACE TOCOPY or TORDBA</td>
<td>No</td>
</tr>
<tr>
<td>REORG INDEX SHRLEVEL NONE, REFERENCE, or CHANGE</td>
<td>No</td>
</tr>
<tr>
<td>REORG TABLESPACE SHRLEVEL NONE UNLOAD CONTINUE or PAUSE, REORG SHRLEVEL REFERENCE, or REORG SHRLEVEL CHANGE</td>
<td>No</td>
</tr>
</tbody>
</table>
### Table 71. Compatibility of REORG INDEX with other utilities (continued)

<table>
<thead>
<tr>
<th>Action</th>
<th>REORG INDEX SHRLEVEL NONE, REFERENCE, or CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>REORG TABLESPACE SHRLEVEL NONE UNLOAD ONLY or EXTERNAL with cluster index</td>
<td>No</td>
</tr>
<tr>
<td>REORG TABLESPACE SHRLEVEL NONE UNLOAD ONLY or EXTERNAL without cluster index</td>
<td>Yes</td>
</tr>
<tr>
<td>REPAIR LOCATE INDEX PAGE REPLACE</td>
<td>No</td>
</tr>
<tr>
<td>REPAIR LOCATE KEY</td>
<td>No</td>
</tr>
<tr>
<td>REPAIR LOCATE RID DELETE</td>
<td>No</td>
</tr>
<tr>
<td>REPAIR LOCATE RID DUMP, VERIFY, or REPLACE</td>
<td>Yes</td>
</tr>
<tr>
<td>REPAIR LOCATE TABLESPACE PAGE REPLACE</td>
<td>Yes</td>
</tr>
<tr>
<td>REPORT</td>
<td>Yes</td>
</tr>
<tr>
<td>RUNSTATS INDEX</td>
<td>No</td>
</tr>
<tr>
<td>RUNSTATS TABLESPACE</td>
<td>Yes</td>
</tr>
<tr>
<td>STOSPACE</td>
<td>Yes</td>
</tr>
<tr>
<td>UNLOAD</td>
<td>Yes</td>
</tr>
</tbody>
</table>

To run on SYSIBM.DSNLUX01 or SYSIBM.DSNLUX02, REORG INDEX must be the only utility in the job step and the only utility that is running in the Db2 subsystem.

### Determining which indexes require reorganization

Reorganizing indexes might improve performance. To determine which indexes to reorganize to potentially gain such a performance improvement, you can analyze certain data in the Db2 catalog. You can then reorganize these indexes by using the REORG INDEX utility.

**Procedure**

To determine which indexes require reorganization:

1. Issue the following SQL statement to identify user-created indexes and Db2 catalog indexes to consider reorganizing with the REORG INDEX utility:

   ```sql
   EXEC SQL
   SELECT IXNAME, IXCREATOR
   FROM SYSIBM.SYSINDEXPART
   WHERE LEAFDIST > 200
   ENDEXEC
   ```

   Using a LEAFDIST value of more than 200 as an indicator of a disorganized index is merely a rough guideline for general cases. This guidance is not absolute. In some cases, 200 is an acceptable value for LEAFDIST. For example, with FREEPAGE 0 and index page splitting, the LEAFDIST value can climb sharply. In this case, a LEAFDIST value that exceeds 200 can be acceptable.
2. Issue the following SQL statement to determine the average distance (multiplied by 100) between successive leaf pages during sequential access of the index.

```sql
EXEC SQL
SELECT LEAFDIST
FROM SYSIBM.SYSINDEXPART
WHERE IXCREATOR = 'index_creator_name'
AND IXNAME = 'index_name'
ENDEXEC
```

An increase in the LEAFDIST value over time probably indicates that the index needs to be reorganized. The optimal value of the LEAFDIST catalog column is zero. However, immediately after you run the REORG and RUNSTATS utilities, LEAFDIST might be greater than zero as a result of empty pages for FREEPAGE and non-leaf pages.

### Using the LEAFDISTLIMIT and REPORTONLY options to determine when reorganization is needed

You can determine when to run REORG for indexes by using the LEAFDISTLIMIT and REPORTONLY options.

**About this task**

**Procedure**

To determine when reorganization is needed:

1. Run the REORG INDEX utility and specify the LEAFDISTLIMIT option and the REPORTONLY option. REORG produces a report with one of the following return codes; but a REORG is not performed.
   1. No limit met; no REORG performed or recommended.
   2. REORG performed or recommended.
2. Optional: Alternatively, information from the SYSINDEXPART catalog table can tell you which indexes qualify for reorganization.

**Related tasks:**

- Maintaining data organization (Db2 Performance)
- Determining when to reorganize indexes (Db2 Performance)

**Related reference:**

- SYSIBM.SYSINDEXPART table (Db2 SQL)

### Access with REORG INDEX SHRLEVEL

You can specify the level of access that you have to your data by using the SHRLEVEL option.

For reorganizing an index or a partition of an index, the SHRLEVEL option lets you choose the level of access that you have to your data during reorganization:
• REORG with SHRLEVEL NONE, the default, reloads the reorganized data into the original area that is being reorganized. Applications have read-only access during unloading and no access during reloading. SHRLEVEL NONE is the only access level that resets REORG-pending status.

• REORG with SHRLEVEL REFERENCE reloads the reorganized data into a new (shadow) copy of the area that is being reorganized. Near the end of reorganization, Db2 switches applications’ future access from the original to the shadow copy. For SHRLEVEL REFERENCE, applications have read-only access during unloading and reloading, and a brief period of no access during switching.

• REORG with SHRLEVEL CHANGE reloads the reorganized data into a shadow copy of the area that is being reorganized. Applications can read from and write to the original area, and Db2 records the writing in the log. Db2 then reads the log and applies it to the shadow copy to bring the shadow copy up to date. This step executes iteratively, with each iteration processing a sequence of log records. Near the end of reorganization, Db2 switches applications’ future access from the original to the shadow copy. Applications have read-write access during unloading and reloading, a brief period of read-only access during the last iteration of log processing, and a brief period of no access during switching.

Log processing with SHRLEVEL CHANGE

When you specify SHRLEVEL CHANGE, Db2 processes the log to update the shadow copy. This step executes iteratively. The first iteration processes the log records that accumulated during the previous iteration. The iterations continue until one of these conditions is met:

• Db2 estimates that the time to perform the log processing in the next iteration will be less than or equal to the time that is specified by MAXRO. If this condition is met, the next iteration is the last.

• Db2 estimates that the switch phase will not start by the deadline specified by DEADLINE. If this condition is met, Db2 terminates reorganization.

• The number of log records that the next iteration will process is not sufficiently lower than the number of log records that were processed in the previous iteration. If this condition is met but the first two conditions are not, Db2 sends message DSNU377I to the console. Db2 continues log processing for the length of time that is specified by DELAY and then performs the action specified by LONGLOG.

Operator actions

LONGLOG specifies the action that Db2 is to perform if log processing is not occurring quickly enough. If the operator does not respond to the console message DSNU377I, the LONGLOG option automatically goes into effect. You can take one of the following actions:

• Execute the START DATABASE(db) SPACENAM(ts)... ACCESS(RO) command and the QUIESCE utility to drain the write claim class. Db2 performs the last iteration, if MAXRO is not DEFER. After the QUIESCE, you should also execute the ALTER UTILITY command, even if you do not change any REORG parameters.

• Execute the START DATABASE(db) SPACENAM(ts)... ACCESS(RO) command and the QUIESCE utility to drain the write claim class. Then, after reorganization has made some progress, execute the START DATABASE(db) SPACENAM(ts)... ACCESS(RW) command. This action increases the likelihood
that log processing can improve. After the QUIESCE, you should also execute the **ALTER UTILITY** command, even if you do not change any REORG parameters.

- Execute the **ALTER UTILITY** command to change the value of MAXRO. Changing it to a huge positive value, such as 9999999, causes the next iteration to be the last iteration.
- Execute the **ALTER UTILITY** command to change the value of LONGLOG.
- Execute the **TERM UTILITY** command to terminate reorganization.
- Adjust the amount of buffer space that is allocated to reorganization and to applications. This adjustment can increase the likelihood that log processing improve after adjusting the space, you should also execute the **ALTER UTILITY** command, even if you do not change any REORG parameters.
- Adjust the scheduling priorities of reorganization and applications. This adjustment can increase the likelihood that log processing improve. After adjusting the priorities, you should also execute the **ALTER UTILITY** command, even if you do not change any REORG parameters.

Db2 does not take the action specified in the LONGLOG phrase if any one of these events occurs before the delay expires:
- An **ALTER UTILITY** command is issued.
- A **TERM UTILITY** command is issued.
- Db2 estimates that the time to perform the next iteration is likely to be less than or equal to the time specified on the MAXRO keyword.
- REORG terminates for any reason (including the deadline).

When REORG INDEX is used with SHRLEVEL REFERENCE or SHRLEVEL CHANGE, pending definition changes are materialized for pending alterations on the index. Pending changes are not materialized for pending alterations on the table space. If pending alterations are involved only on the index, advisory-REORG pending status (AREOR) is reset from the index. If REORG INDEX with SHRLEVEL REFERENCE or SHRLEVEL CHANGE is run at the partition level, pending definition changes are not materialized.

REORG INDEX with SHRLEVEL NONE proceeds without materializing pending definition changes if there were any on the index being reorganized.

Index statistics are collected and updated in the Db2 catalog when pending definition changes are materialized during REORG INDEX with SHRLEVEL REFERENCE or SHRLEVEL CHANGE.

---

**Creating a FlashCopy image copy with REORG INDEX**

As part of REORG INDEX processing, you can use FlashCopy technology to quickly take image copies of the target objects.

**About this task**

**Restriction:** You cannot create FlashCopy image copies of indexes that are defined with the COPY NO attribute.

**Procedure**

To create a FlashCopy image copy with REORG INDEX:
Specify FLASHCOPY(YES) or FLASHCOPY(CONSISTENT) in the REORG INDEX utility control statement. Alternatively, you can set the FLASHCOPY_REORG_INDEX subsystem parameter to YES, which specifies that REORG INDEX is to use FLASHCOPY(YES) by default. The value that you specify for the FLASHCOPY option in the REORG INDEX statement always overrides the value for the FLASHCOPY_REORG_INDEX subsystem parameter. Optionally, you can also specify FCCOPYDDN in the REORG INDEX statement. Use this option to specify a template for the FlashCopy image copy. If you do not specify the FCCOPYDDN option in the REORG INDEX statement, the utility uses the value from the FCCOPYDDN subsystem parameter.

Restriction: The data sets that you specify for the FlashCopy image copy must be on FlashCopy Version 2 disk volumes. When you specify FLASHCOPY(YES) or FLASHCOPY(CONSISTENT), REORG INDEX uses FlashCopy technology to create a consistent copy of the target objects. The FlashCopy image copy fails if the FlashCopy Version 2 disk volumes are not available or if any of the other FlashCopy operational restrictions exist. For a list of those operational restrictions, see “FlashCopy image copies” on page 157.

Related concepts:
“FlashCopy image copies” on page 157

Related reference:
- DEFAULT TEMPLATE field (FCCOPYDDN subsystem parameter) (Db2 Installation and Migration)
- REORG INDEX field (FLASHCOPY_REORG_INDEX subsystem parameter) (Db2 Installation and Migration)

### Temporarily interrupting REORG

You can temporarily pause REORG.

If you specify UNLOAD PAUSE, REORG pauses after unloading the index space into the work data set. The job completes with return code 4. You can restart REORG by using the phase restart or current restart. The REORG statement must not be altered.

The SYSIBM.SYSUTIL record for the REORG INDEX utility remains in "stopped" status until REORG is restarted or terminated.

While REORG is interrupted by PAUSE, you can re-define the table space attributes for user defined table spaces. PAUSE is not required for STOGROUP-defined table spaces. Attribute changes are done automatically by a REORG following an ALTER INDEX.

### Improving performance with REORG INDEX

You can improve the performance of the REORG INDEX utility by taking certain actions.

**About this task**

**Recommendation:** Run online REORG during light periods of activity on the table space or index.
Procedure

To improve REORG performance:

- Run REORG concurrently on separate partitions of a partitioned index space. The processor time for running REORG INDEX on partitions of a partitioned index is approximately the same as the time for running a single REORG index job. The elapsed time is a fraction of the time for running a single REORG job on the entire index.

- Schedule REORG with SHRLEVEL CHANGE when the rate of writing is low and transactions are short. Avoid scheduling REORG with SHRLEVEL CHANGE when low-tolerance applications are executing.

- Run REORG with DRAIN_WAIT.

  The DRAIN_WAIT option provides improved control over the time online REORG waits for drains. Also, because the DRAIN_WAIT is the aggregate time that online REORG is to wait to perform a drain on a table space and associated indexes, the length of drains is more predictable than it is when each partition and index has its own individual waiting-time limit.

  By specifying a short delay time (less than the system timeout value, IRLMRWT), you can reduce the impact on applications by reducing timeouts. You can use the RETRY option to provide opportunities for the online REORG INDEX utility to complete successfully. If you do not want to use RETRY processing, you can still use DRAIN_WAIT to set a specific and more consistent limit on the length of drains.

  RETRY allows an online REORG that is unable to drain the objects it requires to try again after a set period (RETRY_DELAY). If the drain fails in the SWITCH phase, the objects remain in their original state (read-only mode for SHRLEVEL REFERENCE or read-write mode for SHRLEVEL CHANGE). Likewise, objects will remain in their original state if the drain fails in the LOG phase.

  Because application SQL statements can queue behind any unsuccessful drain that the online REORG has tried, define a reasonable delay before you try again to allow this work to complete; the default is 5 minutes.

  When the default DRAIN WRITERS is used with SHRLEVEL CHANGE and RETRY, multiple read-only log iterations can occur. Because online REORG can have to do more work when RETRY is specified, multiple or extended periods of restricted access might occur. Applications that run with REORG must perform frequent commits. During the interval between retries, the utility is still active; consequently, other utility activity against the table space and indexes is restricted.

Termination of REORG INDEX

You can terminate the REORG INDEX utility.

If you terminate REORG with the TERM UTILITY command during the UNLOAD phase, objects have not yet been changed, and you can rerun the job.

If you terminate REORG with the TERM UTILITY command during the build phase, the behavior depends on the SHRLEVEL option:

- For SHRLEVEL NONE, the index is left in RECOVER-pending status. After you recover the index, rerun the REORG job.

- For SHRLEVEL REFERENCE or CHANGE, the index keys are reloaded into a shadow index, so the original index has not been affected by REORG. You can rerun the job.
If you terminate REORG with the TERM UTILITY command during the log phase, the index keys are reloaded into a shadow index, so the original index has not been affected by REORG. You can rerun the job.

If you terminate REORG with the TERM UTILITY command during the switch phase, all data sets that were renamed to their shadow counterparts are renamed back, so the objects are left in their original state. You can rerun the job. If a problem occurs in renaming to the original data sets, the objects are left in RECOVER-pending status. You must recover the index.

The REORG-pending status is not reset until the UTILTERM execution phase. If the REORG INDEX utility abnormally terminates or is terminated, the objects are left in RECOVER-pending status.

The following table lists any restrictive states that are set based on the phase in which REORG INDEX terminated.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Effect on restrictive status</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNLOAD</td>
<td>No effect.</td>
</tr>
<tr>
<td>BUILD</td>
<td>Sets REBUILD-pending (RBDP) status at the beginning of the build phase, and resets RBDP at the end of the phase. SHRLEVEL NONE places an index that was defined with the COPY YES attribute in RECOVER pending (RECP) status.</td>
</tr>
<tr>
<td>LOG</td>
<td>No effect.</td>
</tr>
<tr>
<td>SWITCH</td>
<td>Under certain conditions, if TERM UTILITY is issued, it must complete successfully; otherwise, objects might be placed in RECP status or RBDP status. For SHRLEVEL REFERENCE or CHANGE, sets the RECP status if the index was defined with the COPY YES attribute at the beginning of the switch phase, and resets RECP at the end of the phase. If the index was defined with COPY NO, this phase sets the index in RBDP status at the beginning of the phase, and resets RBDP at the end of the phase.</td>
</tr>
</tbody>
</table>

Related reference:
Appendix B, “Advisory or restrictive states,” on page 1037

Restart of REORG INDEX

You can restart a REORG INDEX utility job.

If you restart REORG in the outlined phase, it re-executes from the beginning of the phase. Db2 always uses RESTART(PHASE) by default unless you restart the job in the UNLOAD phase. In this case, Db2 uses RESTART(CURRENT) by default.

If REORG abnormally terminates or a system failure occurs while it is in the UTILTERM phase, you must restart the job with RESTART(PHASE).

The following table provides information about restarting REORG INDEX. For each phase of REORG and for each type of REORG INDEX (with SHRLEVEL NONE, with SHRLEVEL REFERENCE, and with SHRLEVEL CHANGE), the table indicates the types of restart that are allowed (CURRENT and PHASE). None indicates that no restart is allowed. The "Data sets required" column lists the data sets that must exist to perform the specified type of restart in the specified phase.
Table 73. REORG INDEX utility restart information

<table>
<thead>
<tr>
<th>Phase</th>
<th>Type of restart allowed for SHRLEVEL NONE</th>
<th>Type of restart allowed for SHRLEVEL REFERENCE</th>
<th>Type of restart allowed for SHRLEVEL CHANGE</th>
<th>Data sets required</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNLOAD</td>
<td>CURRENT, PHASE</td>
<td>CURRENT, PHASE</td>
<td>None</td>
<td>SYSUT1</td>
<td></td>
</tr>
<tr>
<td>BUILD</td>
<td>CURRENT, PHASE</td>
<td>CURRENT, PHASE</td>
<td>None</td>
<td>SYSUT1</td>
<td>1</td>
</tr>
<tr>
<td>LOG</td>
<td>Phase does not occur</td>
<td>Phase does not occur</td>
<td>None</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>SWITCH</td>
<td>Phase does not occur</td>
<td>CURRENT, PHASE</td>
<td>CURRENT, PHASE</td>
<td>originals and shadows</td>
<td>1</td>
</tr>
</tbody>
</table>

Note:
1. You can restart the utility with either RESTART or RESTART(PHASE). However, because this phase does not take checkpoints, RESTART always re-executes from the beginning of the phase.

If you restart a REORG STATISTICS job that was stopped in the BUILD phase by using RESTART CURRENT, inline statistics collection does not occur. To update catalog statistics, run the RUNSTATS utility after the restarted job completes. Restarting a REORG STATISTICS job with RESTART(PHASE) is conditional after executing UNLOAD PAUSE. To determine if catalog table statistics are to be updated when you restart a REORG STATISTICS job, see the following table. This table lists whether or not statistics are updated based on the execution phase and whether the job is restarted with RESTART(CURRENT) or RESTART(PHASE).

Table 74. Whether statistics are updated when REORG INDEX STATISTICS jobs are restarted in certain phases

<table>
<thead>
<tr>
<th>Phase</th>
<th>RESTART CURRENT</th>
<th>RESTART PHASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTILINIT</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>UNLOAD</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>BUILD</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Related tasks:
“Restarting an online utility” on page 44
“Restarting a utility after the output data set is full” on page 48

Review of REORG INDEX output

The output from REORG INDEX consists of a reorganized index or index partition.

The following table summarizes the results of REORG INDEX based upon what you specified.

Table 75. Summary of the results of REORG INDEX

<table>
<thead>
<tr>
<th>Specification</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>REORG INDEX</td>
<td>Entire index (all partitions of a partitioned index)</td>
</tr>
<tr>
<td>REORG INDEX PART n</td>
<td>Part n of partitioned index</td>
</tr>
</tbody>
</table>

When reorganizing an index, REORG leaves free pages and free space on each page in accordance with the current values of the FREEPAGE and PCTFREE parameters. (You can set those values by using the CREATE INDEX or ALTER INDEX statement.) REORG leaves one free page after reaching the FREEPAGE limit for each table in the index space.
Catalog updates: REORG INDEX updates SYSINDEXPART OLDEST_VERSION and SYSINDEXES OLDEST_VERSION (if applicable).

Effect of REORG INDEX on index version numbers

Db2 stores the range of used index version numbers in the OLDEST_VERSION and CURRENT_VERSION columns of the SYSIBM.SYSINDEXES and SYSIBM.SYSINDEXPART catalog tables.

The OLDEST_VERSION column contains the oldest used version number, and the CURRENT_VERSION column contains the current version number.

When you run REORG INDEX, the utility updates this range of used version numbers for indexes that are defined with the COPY NO attribute. REORG INDEX sets the OLDEST_VERSION column to the current version number, which indicates that only one version is in use; Db2 can then reuse all of the other version numbers.

Recycling of version numbers is required when all of the version numbers are being used. All version numbers are being used when one of the following situations is true:

- The value in the CURRENT_VERSION column is one less than the value in the OLDEST_VERSION column.
- The value in the CURRENT_VERSION column is 15 and the value in the OLDEST_VERSION column is 0 or 1.

You can also run LOAD REPLACE, REBUILD INDEX, or REORG TABLESPACE to recycle version numbers for indexes that are defined with the COPY NO attribute. To recycle version numbers for indexes that are defined with the COPY YES attribute or for table spaces, run MODIFY RECOVERY.

Related concepts:

- Table space versions (Db2 Administration Guide)

Sample REORG INDEX control statements

Use the sample control statements as models for developing your own REORG INDEX control statements.

Example 1: Reorganizing an index

The following control statement specifies that the REORG INDEX utility is to reorganize index XMSGTXT1. The UNLOAD PAUSE option indicates that after the data has been unloaded, the utility is to stop. Processing can be restarted in the RELOAD phase. This option is useful if you want to redefine data sets during reorganization.

```
REORG INDEX DSN8B10.XMSGTXT1
  UNLOAD PAUSE
```
Example 2: Collecting inline statistics while reorganizing an index.

The following control statement specifies that REORG INDEX is to collect statistics for index XEMPL1 while reorganizing that index. The SHRLEVEL REFERENCE option indicates that during this processing, only read access is allowed on the areas that are being reorganized.

```
REORG INDEX DSN8B10.XEMPL1
          SHRLEVEL REFERENCE STATISTICS
```

Example 3: Updating access path statistics in the catalog and catalog history tables while reorganizing an index

The following control statement specifies that while reorganizing index IU0E0801, REORG INDEX is to collect statistics and update access path statistics in the catalog and catalog history tables. The utility is also to send any output, including space and access path statistics, to SYSPRINT.

```
REORG INDEX IU0E0801
          STATISTICS
          REPORT YES
          UPDATE ACCESSPATH
          HISTORY ACCESSPATH
```

Example 4: Reorganizing a list of indexes

In the following control statement, the OPTIONS utility control statement specifies that the subsequent TEMPLATE and LISTDEF utility control statements are to run in PREVIEW mode. If the syntax of these statements is correct, Db2 expands the REORG_INDEX list and the data set names in the SREC, SUT1, and SOUT templates and prints these results to the SYSPRINT data set. The second OPTIONS control statement turns off the PREVIEW mode, and the subsequent REORG INDEX job runs normally.

The REORG INDEX statement specifies that the utility is to reorganize the indexes that are included in the REORG_INDEX list. The SHRLEVEL CHANGE option indicates that during this processing, read and write access is allowed on the areas that are being reorganized, with the exception of a 100-second period during the last iteration of log processing. During this time, which is specified by the MAXRO option, applications have read-only access. The WORKDDN option indicates that REORG INDEX is to use the data set that is defined by the SUT1 template. If the SWITCH phase does not begin by the deadline that is specified on the DEADLINE option, processing terminates.
Example 5: Creating a FlashCopy image copy with REORG INDEX

The following REORG INDEX control statement reorganizes the index spaces associated with table space DSN8S81E and creates a FlashCopy image copy of the index.

//SYSADMA JOB (ACCOUNT),'NAME',NOTIFY=&SYSUID
//*
//UTIL EXEC DSNUPROC,SYSTEM=VA1A,UID='TEMP',UTPROC=''
//DSNUPROC.SYSREC DD DSN=SYSOPS.DSNAME,
// DISP=(NEW,DELETE),
// SPACE=(CYL,(20,20),RLSE),
// UNIT=SYSDA,VOL=SER=SCR03
//DSNUPROC.SYSUT1 DD DSN=SYSOPS.SYSUT1,
// DISP=(NEW,DELETE,DELETE),
// SPACE=(CYL,(9,90),RLSE),
// UNIT=SYSDA,VOL=SER=SCR03
//DSNUPROC.SYSIN DD *
LISTDEF COPY_LIST INCLUDE INDEX DSN8S81E
TEMPLATE SCOPY UNIT(SYSDA) DISP(NEW,CATLG,DELETE)
DSN(DSN8S81E.PARTLEVEL ALL)
TEMPLATE FCOPY UNIT(SYSDA) DISP(NEW,CATLG,DELETE)
DSN(DSN8S81E.PARTLEVEL ALL)
REORG INDEX LIST COPY_LIST SHRLEVEL REFERENCE FLASHCOPY YES
FCCOPYDNN(SCOPY)

Example 6: Reorganizing clone indexes

The following control statement specifies that REORG INDEX is to reorganize only the specified index spaces that contain indexes on clone tables. The SHRLEVEL CHANGE option indicates that during this processing, applications can read and write to the area.

REORG INDEX ADMF001.IPJM0901 SHRLEVEL CHANGE CLONE

Figure 59. Example statements for job that reorganizes a list of indexes
Chapter 28. REORG TABLESPACE

The REORG TABLESPACE online utility reorganizes a table space, partition, or range of partitions to reclaim fragmented space and improve access performance. You can also run REORG TABLESPACE to materialize pending definition changes.

You can use the DSNACCOX stored procedure to get recommendations on when to run REORG TABLESPACE.

To avoid the cost of running the RUNSTATS utility after running REORG TABLESPACE, you can request that REORG collect inline statistics by specifying the STATISTICS option.

You can run REORG TABLESPACE on the table spaces in the Db2 catalog database (DSNDB06) and on some table spaces in the directory database (DSNDB01). You cannot run REORG TABLESPACE on any table space in the DSNDB07 database.

Do not execute REORG on an object if another member holds retained locks on the object or has long-running non-committing applications that use the object. You can use the DISPLAY GROUP command to determine whether a member status is failed. You can use the DISPLAY DATABASE command with the LOCKS option to determine whether locks are held.

If you issue REORG TABLESPACE with the INITCDDS YES option, REORG does not reorganize the table space. REORG copies the existing compression dictionaries for a table space that is defined with DATA CAPTURE CHANGES into the compression dictionary data set (CDDS). This option applies only to a data sharing group in a GDPS® Continuous Availability with zero data loss (GDPS Continuous Availability with zero data loss) environment.

You can use the REORG TABLESPACE utility in conjunction with z/OS DFSMS data set encryption to encrypt or decrypt table spaces and indexes. REORG TABLESPACE always allocates new Db2-managed data sets unless the REUSE option is specified.

Output

The following table summaries the results of REORG TABLESPACE according to the type of REORG specified.

<table>
<thead>
<tr>
<th>Type of REORG specified</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>REORG TABLESPACE</td>
<td>Reorganizes all data and all indexes.</td>
</tr>
<tr>
<td>REORG TABLESPACE PART n</td>
<td>Reorganizes data for PART n of the table space and PART n of all partitioned indexes.</td>
</tr>
<tr>
<td>REORG TABLESPACE PART n:m</td>
<td>Reorganizes data for PART n through PART m of the table space and PART n through PART m of all partitioned indexes.</td>
</tr>
</tbody>
</table>

© Copyright IBM Corp. 1983, 2017
Table 76. Summary of REORG TABLESPACE output (continued)

<table>
<thead>
<tr>
<th>Type of REORG specified</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note: When SCOPE PENDING is also specified, the REORG TABLESPACE utility reorganizes the specified table space only if it is in REORG-pending or advisory REORG-pending status. For a partitioned table space, REORG TABLESPACE SCOPE PENDING reorganizes only the partitions that are in REORG-pending or advisory REORG-pending status.</td>
<td></td>
</tr>
</tbody>
</table>

If the table space or partition has the COMPRESS YES attribute, the data is compressed when it is reloaded. If you specify the KEEPDICTIONARY option of REORG, the current dictionary is used; otherwise a new dictionary is built. If a table has DATA CAPTURE CHANGES active, any previously existing dictionary is written to the log.

REORG TABLESPACE materializes pending limit key changes if you specify SHRLEVEL REFERENCE or CHANGE.

Authorization required

To execute this utility on a user table space, you must use a privilege set that includes one of the following authorities:
- REORG privilege for the database
- DBADM or DBCTRL authority for the database. If the object on which the utility operates is in an implicitly created database, DBADM authority on the implicitly created database or DSNDB04 is required.
- SYSCTRL authority
- SYSADM authority
- DATAACCESS authority

To execute this utility on a table space in the catalog or directory, you must use a privilege set that includes one of the following authorities:
- REORG privilege for the DSNDB06 (catalog) database
- DBADM or DBCTRL authority for the DSNDB06 (catalog) database
- Installation SYSOPR authority
- SYSCTRL authority
- SYSADM or Installation SYSADM authority
- STATS privilege for the database is required if STATISTICS keyword is specified.

If you specify REORG TABLESPACE SHRLEVEL CHANGE and you create a mapping table, you must use a privilege set that includes DELETE, INSERT, and UPDATE privileges on the mapping table.

If either the FLASHCOPY YES or FLASHCOPY CONSISTENT option is specified, the user ID that invokes the REORG TABLESPACE utility must have the authority to execute the DFSMSdss COPY command.

To run REORG TABLESPACE STATISTICS REPORT YES, you must use a privilege set that includes the SELECT privilege on the catalog tables and tables for which statistics are to be gathered.

An authority other than installation SYSADM or installation SYSOPR can receive message DSNT5001 resource unavailable, while trying to reorganize a table space.
in the catalog or directory. This message can be issued when the
DSNDB06.SYSDBAUT or DSNDB06.SYSUSER catalog table space or one of the
indexes is unavailable. If this problem occurs, run the REORG TABLESPACE utility
again using an authorization ID with the installation SYSADM or installation
SYSOPR authority.

If you use RACF access control with multilevel security and REORG TABLESPACE
is to process a table space that contains a table that has multilevel security with
row-level granularity, you must be identified to RACF and have an accessible valid
security label. You must also meet the following authorization requirements:

- For REORG statements that include the UNLOAD EXTERNAL option, each row
  is unloaded only if your security label dominates the data security label. If your
  security label does not dominate the data security label, the row is not unloaded,
  but Db2 does not issue an error message.
- For REORG statements that include the DISCARD option, qualifying rows are
discarded only if one of the following situations are true:
  - Write-down rules are in effect, you have write-down privilege, and your
    security label dominates the data’s security label.
  - Write-down rules are not in effect and your security label dominates the
    data’s security label.
  - Your security label is equivalent to the data security label.

**Execution phases of REORG TABLESPACE**

The REORG TABLESPACE utility operates in these phases:

**UTILINIT**
Performs initialization and setup.

**UNLOAD**
Unloads the table space and sorts data if a clustering index exists and the
utility job does not include the SORTDATA NO options. For SORTDATA
processing, if you specify NOSYSREC, the utility passes rows in memory
to the RELOAD phase; otherwise, it writes them to a sequential data set.

Nonpartitioned indexes are processed in one of two ways:

- If PART SHRLEVEL REFERENCE or PART SHRLEVEL CHANGE is
  specified, during UNLOAD one or more subtasks unload nonpartitioned
  indexes and build shadow nonpartitioned indexes.
- If PART SHRLEVEL REFERENCE or CHANGE is specified and
  SORTNPSI YES or AUTO is specified or subsystem parameter
  REORG_PART_SORT_NPSI is enabled, during UNLOAD one or more
  subtasks processes nonpartitioned secondary index keys from parts that
  are not within the scope of the REORG. These keys are routed to a sort
  process to be sorted with the keys from parts within the scope of the
  REORG. The shadow index is built from this sorted set of keys.

**RELOAD**
Reloads data from the sequential data set into the table space and creates
full image copies if you specify COPYDDN, RECOVERYDDN, SHRLEVEL
REFERENCE, or SHRLEVEL CHANGE. A subtask sorts the index keys.
The utility also updates table and table space statistics.

**SORT**
Sorts index keys. The sorted keys are passed in memory to the BUILD
phase.
BUILD
Builds indexes and updates index statistics.

SORTBLD
If parallel index build occurs, all activities that normally occur in both the
SORT and BUILD phases occur in the SORTBLD phase instead.

LOG
Processes the log iteratively and appends changed pages to the full image
copies. This phase occurs only if you specify SHRLEVEL CHANGE or
SHRLEVEL REFERENCE PART x.

SWITCH
Switches access to shadow copy of table space or partition. This phase
occurs only if you specify SHRLEVEL REFERENCE or CHANGE.

UTILTERM
Performs cleanup.

Execution phases of REORG TABLESPACE on a LOB table space

The REORG TABLESPACE utility operates in these phases when you run it on a
LOB table space:

<table>
<thead>
<tr>
<th>Phase</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTILINIT</td>
<td>Performs initialization and setup.</td>
</tr>
<tr>
<td>REORGLOB</td>
<td>For SHRLEVEL REFERENCE, the utility unloads LOBs to a shadow data set. RECOVER-pending is not set on the LOB table space. Any error during this phase leaves the original data set intact.</td>
</tr>
<tr>
<td>SWITCH</td>
<td>Switches access to shadow copy of table space or partition.</td>
</tr>
<tr>
<td>UTILTERM</td>
<td>Performs cleanup.</td>
</tr>
</tbody>
</table>

You cannot restart REORG TABLESPACE on a LOB table space in the REORGLOB phase. Before executing REORG TABLESPACE SHRLEVEL NONE on a LOB table space that is defined with LOG NO, you should take a full image copy to ensure recoverability. For SHRLEVEL REFERENCE, an inline image copy is required to ensure recoverability.

Related tasks:
- Maintaining data organization (Db2 Performance)
- Collecting statistics by using Db2 utilities (Db2 Performance)
- Reducing the cost of collecting statistics (Db2 Performance)

Syntax and options of the REORG TABLESPACE control statement

The REORG TABLESPACE utility control statement, with its multiple options, defines the function that the utility job performs.

You can create a control statement with the ISPF/PDF edit function. After creating it, save it in a sequential or partitioned data set. When you create the JCL for running the job, use the SYSIN DD statement to specify the name of the data set.
that contains the utility control statement. Some of these options are not valid for LOB table spaces. For a list of those options, see “Reorganization of a LOB table space” on page 644.
Syntax diagram

Notes:
1. The default for SORTNPSI is the value of the REORG_PART_SORT_NPSI subsystem parameter.
2. You cannot use UNLOAD PAUSE with the LIST option.
**copy-spec:**

```
COPYDDN(SYSCOPY)
COPYDDN(ddname1, ddname2)
RECOVERYDDN(ddname3, ddname4)
FLASHCOPY
FLASHCOPY_YES
FLASHCOPY_CONSISTENT
FCCOPYDDN(template-name)
```

**Notes:**

1. COPYDDN(SYSCOPY) is not the default if you specify SHRLEVEL NONE, and no partitions are in REORG-pending status.
2. Either COPYDDN or FCCOPYDDN can be specified, or they can both be specified. At least one of these options must be specified for SHRLEVEL NONE when a partition is in REORG-pending status, or for SHRLEVEL CHANGE OR SHRLEVEL REFERENCE. If you specify FCCOPYDDN, but do not specify COPYDDN or a SYSCOPY DD statement or TEMPLATE specification, for SYSCOPY, only a FlashCopy image copy is taken.

**SHRLEVEL-REFERENCE-spec:**

```
SHRLEVEL REFERENCE deadline-spec drain-spec change-spec
FASTSWITCH_YES AUX NO
FASTSWITCH_NO AUX YES
```

**Notes:**

1. For SHRLEVEL REFERENCE, the change-spec options are ignored. The exception is a partition-level REORG operation on a partitioned table space that has a non-partitioned index. In that case, the change-spec options are not ignored.

**SHRLEVEL-CHANGE-spec:**
**deadline-spec:**

- DEADLINE: NONE
- DEADLINE: timestamp
  - labeled-duration-expression

**drain-spec:**

1. DRAIN_WAIT: integer
2. RETRY: integer
3. RETRY_DELAY: integer
4. TIMEOUT: TERM
5. TIMEOUT: ABEND
6. LOGRANGES: YES
7. LOGRANGES: NO
8. DRAIN_ALLPARTS: NO
9. DRAIN_ALLPARTS: YES
10. SWITCHTIME: NONE
11. SWITCHTIME: timestamp
   - labeled-duration-expression
12. NEWMAXRO: NONE
13. NEWMAXRO: integer

**Notes:**
1. The default for DRAIN_WAIT is the value of the IRLMRWT subsystem parameter.
2. The default for RETRY is the value of the UTIMOUT subsystem parameter.
3. The default for RETRY_DELAY is the smaller of the following two values: DRAIN_WAIT value \times RETRY value, DRAIN_WAIT value \times 10

**change-spec:**

For SHRLEVEL REFERENCE, the change-spec options are ignored. The exception is a partition-level REORG operation on a partitioned table space that has a non-partitioned index. In that case, the change-spec options are not ignored.
Notes:
1  The default for MAXRO is the RETRY_DELAY default value.
STATISTICS

stat-table-spec

- TABLE—(ALL—)
- SAMPLE—integer
- TABLE—(table-name)

table-stats-spec:

- COLUMN—ALL
- SAMPLE—integer
- COLUMN—(column-name)

- COLGROUP—(column-name)

colgroup-stats-spec:

- FREQVAL—COUNT—MOST
- COUNT—integer
- MOST—
- BOTH—
- LEAST—

- NUMQUANTILES—integer
- HISTOGRAM
- NUMQUANTILES—integer

stat-index-spec
correlation-stats-spec:

(1)

KEYCARD

FREQVAL

NUMCOLS

COUNT

10

HISTOGRAM

NUMCOLS

NUMQUANTILES

100

Notes:
1. The KEYCARD option is deprecated. The KEYCARD functionality is now built into the default execution of the inline statistics for indexes and cannot be disabled.

FROM-TABLE-spec:

FROM

TABLE

table-name

WHEN

(selection-condition-spec)

selection-condition-spec:

predicate

selection condition

AND

OR

predicate

selection condition

predicate:
basic predicate:

```
基本 predicate:
```

```
column-name  =  constant
<=
>=

Notes:

1. The following forms of the comparison operators are also supported in basic and quantified predicates: !=, !<, and !>. For details, see comparison operators.

BETWEEN predicate:

```
BETWEEN predicate:
```

```
constant  BETWEEN  labeled-duration-expression

IN predicate:

```
IN predicate:
```

```
constant  IN  (constant)

LIKE predicate:

```
LIKE predicate:
```

```
column-name  LIKE  string-constant

564 Utility Guide and Reference
NULL predicate:

```
column-name IS NOT NULL
```

discard-spec:

```
DISCARD NOPAD YES
```

INITCDDS-spec:

```
INITCDDS NO
```

Option descriptions

**TABLESPACE database-name.table-space-name**

Specifies the table space (and, optionally, the database to which it belongs) that is to be reorganized.

If you reorganize a table space, its indexes are also reorganized.

**database-name**

Is the name of the database to which the table space belongs. The name cannot be DSNDB07.

The default value is **DSNDB04**.

**table-space-name**

Is the name of the table space that is to be reorganized. The name cannot be SYSUTILX if the specified database name is DSNDB01.

**LIST listdef-name**

Specifies the name of a previously defined LISTDEF list name. The utility allows one LIST keyword for each control statement of REORG TABLESPACE. The list must contain only table spaces.

Do not specify FROM TABLE, STATISTICS TABLE table-name, or STATISTICS INDEX index-name with REORG TABLESPACE LIST. If you want to collect inline statistics for a list of table spaces, specify STATISTICS TABLE (ALL). If you want to collect inline statistics for a list of indexes, specify STATISTICS INDEX (ALL). Do not specify PART with LIST.
REORG TABLESPACE is invoked once for each item in the list. This utility will only process clone data if the CLONE keyword is specified. The use of CLONED YES on the LISTDEF statement is not sufficient.

The partitions or partition ranges can be specified in a list.

LISTPARTS \( n \)

Specifies the maximum number of data partitions to be reorganized in a single REORG on a LISTDEF that contains PARTLEVEL list items.

\( n \) Specifies an integer that represents the maximum number of data partitions to be reorganized at once. Valid value is greater than 0. If LISTPARTS is not specified, the default value is the setting of the REORG_LIST_PROCESSING subsystem parameter.

PARALLEL(\( num\)-subtasks)

Specifies the maximum number of subtasks that are to be started in parallel to reorganize a table space. If the PARALLEL keyword is omitted, the maximum number of subtasks is limited by either the number of partitions that are being unloaded or the number of indexes that are built.

The value of \( num\)-subtasks must be an integer between 0 and 32767, inclusive. If the specified value for \( num\)-subtasks is greater than 32767, the REORG TABLESPACE statement fails. If 0 or no value is specified for \( num\)-subtasks, the REORG TABLESPACE utility uses the optimal number of parallel subtasks. If the specified value for \( num\)-subtasks is greater than the calculated optimal number, the REORG TABLESPACE utility limits the number of parallel subtasks to the optimal number with applied constraints.

The specified number of subtasks for PARALLEL always overrides the specification of the PARAMDEG_UTIL subsystem parameter, so PARALLEL can be smaller or larger than the value of PARAMDEG_UTIL.

REORG TABLESPACE uses sophisticated algorithms to allocate subtasks for unloading partitions, reloading partitions, building indexes, applying log changes, and gathering statistics. As a result, the number of subtasks that are started might be less than the number specified on PARALLEL.

INITCDDS

Specifies whether REORG TABLESPACE copies the existing compression dictionaries for a table space that is defined with DATA CAPTURE CHANGES into the compression dictionary data set (CDDS), instead of reorganizing the table space. This option applies only to a data sharing group in a GDPS Continuous Availability with zero data loss (GDPS Continuous Availability with zero data loss) environment.

NO Specifies that REORG TABLESPACE performs normal REORG processing.

YES Specifies that REORG TABLESPACE populates the CDDS, and does not reorganize the table space. Specify YES only when the tables in the table space are defined with DATA CAPTURE CHANGES, the table space is in a data sharing group, and the data sharing group is part of a GDPS Continuous Availability with zero data loss environment.

SEARCHTIME

Specifies the timestamp that REORG TABLESPACE uses to determine which expansion dictionary to use to populate the CDDS. If the expansion dictionary that the target page set is currently using has an LRSN value that is later than the SEARCHTIME value, REORG searches for the prior expansion dictionary in the Db2 log and in
SYSCOPY information. If a prior expansion dictionary exists, REORG copies that dictionary into the CDDS.

timestamp or the result of evaluation of labeled-duration-expression must be a TIMESTAMP(6) value.

NONE
  Specifies that REORG populates the CDDS from the expansion dictionary that the target page set is currently using.

timestamp
  Specifies a TIMESTAMP(6) constant that determines which expansion dictionary to use to populate the CDDS.

labeled-duration-expression
  Evaluates to a TIMESTAMP(6) constant that determines which expansion dictionary to use to populate the CDDS. The calculation is based on either CURRENT TIMESTAMP or CURRENT DATE. You can add or subtract one or more constant values to or from the CURRENT TIMESTAMP or CURRENT DATE value. The expression must not evaluate to a timestamp that occurs after REORG is run. CURRENT TIMESTAMP and CURRENT DATE are evaluated once, when the REORG statement is first processed. If a list of objects is specified, the same value is in effect for all objects in the list.

CURRENT_DATE
  Specifies that the expansion dictionary timestamp is to be calculated based on the CURRENT DATE.

CURRENT_TIMESTAMP
  Specifies that the expansion dictionary timestamp is to be calculated based on the CURRENT TIMESTAMP.

WITH TIME ZONE
  Specifies that the CURRENT TIMESTAMP value includes a time zone. The timestamp precision of the special register CURRENT_TIMESTAMP must be the same as the precision of the timestamp equivalent of the LRSN value for the expansion dictionary. Otherwise the default timestamp precision is used. The time zone of CURRENT_TIMESTAMP is the value of special register CURRENT_TIMEZONE. The comparison is done by comparing the UTC portion of the timestamps.

constant
  Indicates a unit of time and is followed by one of the seven duration keywords: YEARS, MONTHS, DAYS, HOURS, MINUTES, SECONDS, or MICROSECONDS. The singular form of these words is also acceptable: YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, MICROSECOND.

CLONE
  Indicates that REORG TABLESPACE is to reorganize only clone tables from the specified table spaces. This utility will only process clone data if the CLONE keyword is specified. The use of CLONED YES on the LISTDEF statement is not sufficient. Base tables in the specified table spaces are not reorganized. If you specify CLONE, you cannot specify STATISTICS. Statistics are not collected for clone tables.
REUSE
When used with SHRLEVEL NONE, specifies that REORG is to logically reset and reuse Db2-managed data sets without deleting and redefining them. If you do not specify REUSE and SHRLEVEL NONE, Db2 deletes and redefines Db2-managed data sets to reset them.

If a data set has multiple extents, the extents are not released if you use the REUSE parameter.

REUSE does not apply if you also specify SHRLEVEL REFERENCE or CHANGE.

SCOPE
Indicates the scope of the reorganization of the specified table space or of one or more specified partitions.

ALL
Indicates that you want the specified table space or one or more partitions to be reorganized. The default is ALL.

PENDING
Indicates that you want the specified table space or one or more partitions to be reorganized only if they are in REORG-pending (REORP, AREO*, or AREOR) status.

PART(integer)
PART(integer1:integer2)
PART(integer,...integer,...integer1:integer2,...integer1:integer2)
Identifies the set of partitions that are to be reorganized. The set of partitions must be enclosed in parentheses.

You can reorganize:
• One or more single partitions
• One or more ranges of partitions
• A combination of one or more single partitions and one or more ranges of partitions

The partitions do not need to be consecutive.

integer, integer1, and integer2 are physical partition numbers. Each partition number must be in the range from 1 to the number of partitions that are defined for the table space or partitioning index. The maximum is 4096.

integer
Designates a single partition.

integer1:integer2
Designates a range of existing table space partitions. You can specify the range in any order. For example, you can specify (100:199) or (199:100). In either case, partitions 100 through 199 are reorganized. If integer1 is equal to integer2, only that one partition is reorganized. For example, if you specify (5:5), only partition 5 is reorganized.

If you omit the PART keyword, the entire table space is reorganized.

If you specify the PART keyword for a LOB table space, Db2 issues an error message, and utility processing terminates with return code 8.

If you specify a partition range and the high or low partitions in the list are in a REORG-pending state, the adjacent partition that is outside the specified range must not be in REORG-pending state; otherwise, the utility terminates with an error.
Restriction: You cannot run concurrent REORG TABLESPACE SHRLEVEL CHANGE PART integer on the same table space with one or more non-partitioned indexes defined in it. Instead of submitting multiple jobs, you can merge the jobs into one job by specifying all the target partitions in the same REORG job.

REBALANCE
Specifies that REORG TABLESPACE is to set new partition boundaries so that rows are evenly distributed across the reorganized partitions. If the columns that are used in defining the partition boundaries have many duplicate values within the data rows, even balancing is not always possible. Specify REBALANCE for more than one partition; if you specify a single partition for rebalancing, REORG TABLESPACE ignores the specification.

You can specify REBALANCE with SHRLEVEL NONE, SHRLEVEL CHANGE, or SHRLEVEL REFERENCE if the base table space has an associated auxiliary LOB table space. In this case, you must also specify AUX YES, which is the default value if you specify REBALANCE. When REBALANCE is specified with SHRLEVEL REFERENCE or SHRLEVEL CHANGE, pending definition changes for conversion of a partitioned table space to a partition-by-range table space are not materialized.

REBALANCE cannot be specified with SCOPE PENDING.

Restrictions: REBALANCE cannot be specified for the following objects:
- Partition-by-growth table spaces
- Base tables with XML columns
- XML table spaces
- An object that is involved in a clone relationship. (Because the base and clone tables share catalog information, REBALANCE can change the partition boundaries of the target table.)
- Table spaces with pending alter limit key changes

When you specify REBALANCE, you must create an inline copy by performing one of the following actions:
- Provide a SYSCOPY DD statement in the JCL.
- Use the TEMPLATE utility to dynamically allocate the SYSCOPY data set.
- Specify a DD name with the COPYDDN option in the REORG control statement and specify either a corresponding DD statement or TEMPLATE statement.

At completion, Db2 invalidates packages and the dynamic cache.

SORTCLUSTER
Determines whether REBALANCE is to attempt to sort the data records into clustering order. This option is ignored if no clustering index exists in the table, or when the limit key columns are identical to or are a superset of the clustering index columns.

NO Specifies that the data records are not to be explicitly sorted into clustering order. This option is the default behavior.

If SORTCLUSTER NO is explicitly specified, REORG-pending advisory (AREO*) status is not set on the affected data partitions upon REORG REBALANCE completion. If SORTCLUSTER NO is not explicitly specified and instead accepted by default, AREO* status can still be set on the affected partitions.
YES
Specifies that the data records are to be explicitly sorted into clustering order as needed.

LOG
Specifies whether records are to be logged during the RELOAD phase of REORG SHRLEVEL NONE. If the records are not logged, the table space is recoverable only after an image copy is taken. An image copy is taken during the REORG execution if COPYDDN, FCCOPYDDN, RECOVERYDDN, SHRLEVEL REFERENCE, or SHRLEVEL CHANGE is specified.

YES
Specifies that log records are to be taken during the RELOAD phase. This option is not allowed for any table space in DSND01 or DSND06, or if the SHRLEVEL REFERENCE or CHANGE option is specified.

The default value is YES if SHRLEVEL NONE is specified explicitly or by default. If SHRLEVEL NONE is specified but the table space has the NOT LOGGED attribute, Db2 processes LOAD with LOG NO.

NO
Specifies that records are not to be logged. This option is the default and required if the SHRLEVEL REFERENCE or CHANGE option is specified. LOG NO puts the table space in COPY-pending status when REORG is executed remotely and RECOVERYDDN is not specified.

SORTDATA or SORTDATA NO
SORTDATA specifies that the data is to be unloaded by a table space scan, and sorted in clustering order.

The default value is SORTDATA, unless you specify UNLOAD ONLY or UNLOAD EXTERNAL. If you specify one of these options, the default is SORTDATA NO.

SORTDATA NO specifies that, when possible, the data is to be unloaded in the order of the clustering index. Specify SORTDATA NO if one of the following conditions is true:
- The data is in or near perfect clustering order, and the REORG utility is used to reclaim space from dropped tables.
- The amount of data is very large, and an insufficient amount of disk space is available for sorting.

For a partitioned table space, REORG does not unload the records by way of the clustering index when the clustering index is not partitioning. The data records must be unloaded by partition order first. In addition, when REORG unload or reload partition parallelism is used, or when REORG is run on a partition-by-growth table space, REORG always performs a table space scan to unload the data records, when the clustering index is not used.

Related information:
- Clustering indexes (Introduction to Db2 for z/OS)

RECLUSTER
Specifies whether data records are to be reclustered by unloading them by way of the clustering index. For SHRLEVEL CHANGE processing, or when the clustering index has the EXCLUDE NULL KEYS attribute, RECLUSTER NO is always enforced.

YES
Data records are to be reclustered and to be unloaded by the clustering index if one exists. This option is the default behavior.
NO  Data records are not to be reclustered and are not unloaded by way of the clustering index.

NOSYSREC
Specifies that REORG TABLESPACE is not to use an unload data set. The utility uses the output of sorting as the input to reload but does not use an unload data set for this process.

Omitting the unload data set can improve performance. However, when you omit the unload data set by specifying NOSYSREC, the following limitations exist for restarting the utility:

- If an error occurs during reloading, you must restart the utility at the beginning of the UNLOAD phase. (If you do not specify NOSYSREC, you can start the utility at the RELOAD phase.)
- If you specify SHRLEVEL NONE and an error occurs during reloading, you must first run the RECOVER utility with the most recent image copy.

If you specify SHRLEVEL NONE with NOSYSREC, create an image copy before you run REORG TABLESPACE.

You cannot specify NOSYSREC if any of the following conditions is true:

- No data will be sorted during the REORG TABLESPACE job. Examples:
  - SORTDATA NO is specified, or no index is defined on the data that is being reorganized.
  - UNLOAD PAUSE is specified.
  - UNLOAD ONLY is specified.

COPYDDN (ddname1, ddname2)
Specifies the DD statements for the primary (ddname1) and backup (ddname2) copy data sets for the image copy. ddname1 and ddname2 are the DD names.

The default value is SYSCOPY for the primary copy. A full image copy data set is created when REORG executes. This copy is called an inline copy. The table space does not remain in COPY-pending status regardless of which LOG option you specify.

When an inline copy is performed, Db2 writes a record with ICTYPE='F' in the SYSIBM.SYSCOPY catalog table. The name of the inline copy data set is listed in that record. If an inline copy is performed when REORG is run on a range of partitions, Db2 writes a record with ICTYPE='F' for each partition. The inline copy data set name is the same in all of those records.

If you specify SHRLEVEL NONE (explicitly or by default) for REORG, and COPYDDN is not specified, an image copy is not created at the local site.

COPYDDN(SYSCOPY) is assumed, and a DD statement for SYSCOPY is required if either of the following conditions are true:

- You specify REORG SHRLEVEL REFERENCE or CHANGE, and you do not specify COPYDDN.
- A table space or partition is in REORG-pending (REORP) status.
- You specify REBALANCE.

The COPYDDN keyword specifies either a DD name or a template name from a previous TEMPLATE utility statement. If utility processing detects that the specified name is both a DD name in the current job step and a TEMPLATE
name, the utility uses the DD name. If the template name uses the &PA. or &PART. variable, REORG allocates a separate copy data set for each partition that is being reorganized.

REORG can take inline copies of XML table spaces.

**RECOVERYDDN (ddname3,ddname4)**

Specifies the DD statements for the primary (ddname3) and backup (ddname4) copy data sets for the image copy at the recovery site.

ddname3 and ddname4 are the DD names.

You cannot have duplicate image copy data sets. The same rules apply for RECOVERYDDN as for COPYDDN.

The RECOVERYDDN keyword specifies either a DD name or a template name from a previous TEMPLATE utility statement. If utility processing detects that the specified name is both a DD name in the current job step and a TEMPLATE name, the utility uses the DD name. If the template name uses the &PA. or &PART. variable, REORG allocates a separate copy data set for each partition that is being reorganized.

REORG SHRLEVEL REFERENCE of a LOB table space supports inline copies, but REORG SHRLEVEL NONE does not.

**FLASHCOPY**

Specifies whether FlashCopy technology is used to create a copy of the object. Valid values are YES, NO, or CONSISTENT. When FlashCopy is used, a separate data set is created for each partition or piece of the object.

The FlashCopy specifications on the utility control statement override any specifications for FlashCopy that are defined by using the Db2 subsystem parameters. If the FlashCopy subsystem parameters specify the use of FlashCopy as the default behavior of this utility, the FLASHCOPY option can be omitted from the utility control statement.

**Important:** If the input data set is less than one cylinder, FlashCopy technology might not be used for copying the objects regardless of the FLASHCOPY settings. The copy is performed by IDCAMS if FlashCopy is not used.

**NO** Specifies that no FlashCopy is made. NO is the default value for FLASHCOPY.

**YES** Specifies that FlashCopy technology is used to copy the object.

Specify YES only if the Db2 data sets are on FlashCopy Version 2 disk volumes.

**Important:** Under the following circumstances, the REORG TABLESPACE utility might not use FlashCopy even though YES is specified:

- FlashCopy Version 2 disk volumes are not available
- The source tracks are already the target of a FlashCopy operation
- The target tracks are the source of a FlashCopy operation
- The maximum number of relationships for the copy is exceeded

If FlashCopy is requested but not used, REORG TABLESPACE completes with return code 8. If no sequential inline copy is requested on the same job, the objects are left in COPY-pending status.
CONSISTENT

Specifies that FlashCopy technology is used to copy the object. Because the copies created by the REORG TABLESPACE utility are already consistent, the utility treats a specification of CONSISTENT the same as a specification of YES.

Related information:
“FlashCopy image copies” on page 157
“Subsystem parameters for refining DFSMSdss COPY operation with utilities” on page 40

FCCOPYDDN

Specifies the template to be used to create the FlashCopy image copy data set names. If a value is not specified for FCCOPYDDN on the REORG TABLESPACE control statement when FlashCopy is used, the value specified on the FCCOPYDDN subsystem parameter determines the template to be used.

(template-name)

The data set names for the FlashCopy image copy are allocated according to the template specification. For table space or index space level FlashCopy image copies, because a data set is allocated for each partition or piece, ensure that the data set naming convention in the template specification is unique enough. Use the &DSNUM variable, which resolves to a partition number or piece number at execution time.

AUTOESTSPACE

Specifies that REORG automatically calculates and formats the size of the fixed hash space for hash-organized table spaces. The use of AUTOESTSPACE YES might reduce the number of rows in the overflow area.

YES

Specifies that Db2 uses real-time statistics (RTS) values to adjust the size of the hash space. User-specified HASH SPACE values stored in the SYSIBM.SYSTABLESPACE and SYSIBM.SYSTABLEPART catalog tables are not changed. YES is the default value for AUTOESTSPACE.

NO

Specifies that Db2 uses the HASH SPACE value specified for CREATE TABLE or ALTER TABLE. These values are stored in the SYSIBM.SYSTABLESPACE and SYSIBM.SYSTABLEPART catalog tables.

SHRLEVEL

Specifies the method that is to be used for the reorganization. The parameter that follows SHRLEVEL indicates the type of access that is to be allowed during the RELOAD phase of REORG.

NONE

Specifies that reorganization is to operate as follows:
- Unloading from the area that is being reorganized (while applications can read but cannot write to the area)
- Reloading into that area (while applications have no access), and then allowing read/write access again

If you specify NONE (explicitly or by default), you cannot specify the following parameters:
- MAPPINGTABLE
- MAXRO
- LONGLOG
- DELAY
- DEADLINE
• DRAIN_WAIT
• RETRY
• RETRY_DELAY

Restrictions:
• If you specify UNLOAD PAUSE or UNLOAD ONLY, you cannot specify NOSYSREC. SHRLEVEL NONE cannot be specified for tables that are defined with ORGANIZE BY HASH.
• You cannot specify SHRLEVEL NONE in a REORG TABLESPACE control statement that completes the process of recovery to a point in time prior to the materialization of pending definition changes.

When SHRLEVEL NONE is specified, pending definition changes are not materialized and any associated restrictive states are not reset. For example, pending limit key changes are not materialized and the associated advisory REORG-pending status is not reset. (Immediate alter limit key changes can be materialized by REORG SHRLEVEL NONE.) SHRLEVEL NONE is not supported when REORG is run against a LOB table space.

REFERENCE
Specifies that reorganization is to operate as follows:
• Unloading from the area that is being reorganized (while applications can read but cannot write to the area)
• Reloading into a shadow copy of that area (while applications can read but cannot write to the original copy)
• Switching the future access of an application from the original copy to the shadow copy by exchanging the names of the data sets, and then allowing read/write access again

If you specify SHRLEVEL REFERENCE and FASTSWITCH NO:
• Pending definition changes are not materialized.
• The REORG_DROP_PBG_PARTS subsystem parameter has no effect. If REORG_DROP_PBG_PARTS is set to ENABLE, empty trailing partitions are not removed.

If you specify REFERENCE for a LOB table space, you must take an inline copy during the reorganization.

If you specify REFERENCE, you cannot specify the following parameters:
• LOG. Reorganization with REFERENCE always creates an image copy and always refrains from logging records during reloading.
• UNLOAD PAUSE, UNLOAD ONLY, or UNLOAD EXTERNAL. Reorganization with REFERENCE always uses UNLOAD CONTINUE, which is the default value. (You can explicitly specify UNLOAD CONTINUE or none of the UNLOAD options, but you cannot specify UNLOAD PAUSE, UNLOAD ONLY, or UNLOAD EXTERNAL.)
• MAPPINGTABLE.

Specifying REORG TABLESPACE PART SHRLEVEL REFERENCE with the REORG_PART_SORT_NPSI subsystem parameter enabled might require larger sort work data sets because of the increased number of keys sorted for nonpartitioned secondary indexes.
Specifying SHRLEVEL REFERENCE or CHANGE on an entire XML partitioned table space converts the XML table space to extended 10-byte format if one of the following is true:

- The UTILITY OBJECT CONVERSION subsystem parameter is set to EXTENDED or NOBASIC.
- The RBALRSN_CONVERSION EXTENDED keywords are specified.

**Restriction:** You cannot specify SHRLEVEL REFERENCE when REORG TABLESPACE with PART is run on a NOT LOGGED table space on which nonpartitioned indexes are defined.

**CHANGE**

Specifies that reorganization is to operate as follows:

- Unloading from the area that is being reorganized (while applications can read and write to the area)
- Reloading into a shadow copy of that area (while applications have read/write access to the original copy of the area)
- Applying the log of the original copy to the shadow copy (while applications can read and usually write to the original copy)
- Switching the future access of an application from the original copy to the shadow copy by exchanging the names of the data sets, and then allowing read/write access again

If you specify SHRLEVEL CHANGE and FASTSWITCH NO:

- Pending definition changes are not materialized.
- The REORG_DROP_PBG_PARTS subsystem parameter has no effect. If REORG_DROP_PBG_PARTS is set to ENABLE, empty trailing partitions are not removed.

If you specify CHANGE, you cannot specify the following parameters:

- LOG. Reorganization with CHANGE always creates an image copy and always refrains from logging records during reloading.
- UNLOAD PAUSE, UNLOAD ONLY, or UNLOAD EXTERNAL. Reorganization with CHANGE always uses UNLOAD CONTINUE, which is the default value. (You can explicitly specify UNLOAD CONTINUE or none of the UNLOAD options, but you cannot specify UNLOAD PAUSE, UNLOAD ONLY, or UNLOAD EXTERNAL.)

Performing REORG TABLESPACE PART SHRLEVEL CHANGE with the REORG_PART_SORT_NPSI subsystem parameter enabled might require larger sort work data sets because of the increased number of keys sorted for nonpartitioned secondary indexes.

Specifying SHRLEVEL REFERENCE or CHANGE on an entire XML partitioned table space converts the XML table space to extended 10-byte format if one of the following is true:

- The UTILITY OBJECT CONVERSION subsystem parameter is set to EXTENDED or NOBASIC.
- The RBALRSN_CONVERSION EXTENDED keywords are specified.

If you specify SHRLEVEL CHANGE and do not specify KEEPDICTIONARY when you run REORG TABLESPACE on the source group in a GDPS Continuous Availability with zero data loss environment, Db2 inserts a decompression dictionary into the CDDS during the switch phase.
Restrictions:

- You cannot specify SHRLEVEL CHANGE if the table space has the NOT LOGGED attribute, unless the table space is a LOB table space.
- If you specify SHRLEVEL CHANGE in a REORG TABLESPACE control statement that completes the process of recovery to a point in time prior to the materialization of pending definition changes, REORG issues a message, and uses SHRLEVEL REFERENCE.

DEADLINE

Specifies the deadline for the SWITCH phase to begin. If Db2 estimates that the SWITCH phase will not begin by the deadline, Db2 issues the messages that the DISPLAY UTILITY command would issue and then terminates the reorganization.

The final result and all the timestamp calculation of DEADLINE will be in TIMESTAMP(6).

If REORG SHRLEVEL REFERENCE or SHRLEVEL CHANGE terminates because of a DEADLINE specification, Db2 issues message DSNU374I with reason code 2 but does not set a restrictive status.

NONE

Specifies that a deadline by which the SWITCH phase of log processing must begin does not exist.

timestamp

Specifies the deadline for the SWITCH phase of log processing to begin. This deadline must not have already occurred when REORG is executed.

labeled-duration-expression

Calculates the deadline for the SWITCH phase of log processing to begin. The calculation is based on either CURRENT TIMESTAMP or CURRENT DATE. You can add or subtract one or more constant values to specify the deadline. This deadline must not have already occurred when REORG is executed. CURRENT TIMESTAMP and CURRENT DATE are evaluated once, when the REORG statement is first processed. If a list of objects is specified, the same value will be in effect for all objects in the list.

CURRENT_DATE

Specifies that the deadline is to be calculated based on the CURRENT DATE.

CURRENT_TIMESTAMP

Specifies that the deadline is to be calculated based on the CURRENT TIMESTAMP.

WITH TIME ZONE

Specifies that the CURRENT TIMESTAMP is compared with the time zone column. The timestamp precision of the special register CURRENT_TIMESTAMP should be the same as the column timestamp precision. Otherwise the default timestamp precision is used. The time zone of CURRENT_TIMESTAMP is the value of special register CURRENT_TIMEZONE. The comparison is done by comparing the UTC portion of the timestamp.

constant

Indicates a unit of time and is followed by one of the seven duration keywords: YEARS, MONTHS, DAYS, HOURS, MINUTES, SECONDS,
or MICROSECONDS. The singular form of these words is also acceptable: YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, MICROSECOND.

**DRAIN_WAIT integer**

Specifies the number of seconds that the utility waits when draining the table space or index. The specified time is the aggregate time for objects that are to be reorganized. This value overrides the values that are specified by IRLMRWT and UTIMOUT. Valid values for integer are from 0 to 1800. If the keyword is omitted or if a value of 0 is specified, the utility uses the value of the lock timeout system parameter IRLMRWT.

**RETRY integer**

Specifies the maximum number of retries that REORG is to attempt. Valid values for integer are from 0 to 255.

Specifying RETRY can lead to increased processing costs and can result in multiple or extended periods of read-only access. For example, when you specify RETRY and SHRLEVEL CHANGE, the size of the copy that is taken by REORG might increase.

The default value is the value of the UTIMOUT subsystem parameter.

**RETRY_DELAY integer**

Specifies the minimum duration, in seconds, between retries. Valid values for integer are from 1 to 1800.

If you do not specify RETRY_DELAY, REORG TABLESPACE uses the smaller of the following two values:

- DRAIN_WAIT value × RETRY value
- DRAIN_WAIT value × 10

**MAPPINGTABLE table-name**

Specifies the name of the mapping table that REORG TABLESPACE is to use to map between the RIDs of data records in the original copy of the area and the corresponding RIDs in the shadow copy. Enclose the table name in quotation marks if the name contains a blank. If a mapping table is required, and one is not specified, REORG will create it.

**MAPPINGDATABASE database-name**

Specifies the database in which REORG implicitly creates the mapping table and index objects. This keyword overrides the subsystem parameter value in REORG_MAPPING_DATABASE. The value cannot be DSNDB01, DSNDB06, DSNDB07, implicit database, and work file or temporary database.

**MAXRO integer**

Specifies the maximum amount of time for the last iteration of log processing. During that iteration, applications have read-only access. MAXRO is a log phase parameter. If MAXRO is specified when a log phase is not needed, an error message is issued.

The actual execution time of the last iteration might exceed the specified value for MAXRO.

The ALTER UTILITY command can change the value of MAXRO.

The default value is the RETRY_DELAY default value.

**integer**

integer is the number of seconds. Specifying a small positive value reduces the length of the period of read-only access, but it might increase the
elapsed time for REORG to complete. If you specify a huge positive value, the second iteration of log processing is probably the last iteration.

**DEFER**

Specifies that the iterations of log processing with read-write access can continue indefinitely. REORG never begins the final iteration with read-only access, unless you change the MAXRO value with **ALTER UTILITY**.

If you specify DEFER, you should also specify LONGLOG CONTINUE.

If you specify DEFER, and Db2 determines that the actual time for an iteration and the estimated time for the next iteration are both less than 5 seconds, Db2 adds a 5 second pause to the next iteration. This pause reduces consumption of processor time. The first time this situation occurs for a given execution of REORG, Db2 sends message DSNU362I to the console. The message states that the number of log records that must be processed is small and that the pause occurs. To change the MAXRO value and thus cause REORG to finish, execute the **ALTER UTILITY** command.

Db2 adds the pause whenever the situation occurs; however, Db2 sends the message only if 30 minutes have elapsed since the last message was sent for a given execution of REORG.

**DRAIN**

Specifies drain behavior at the end of the log phase after the MAXRO threshold is reached and when the last iteration of the log is to be applied. DRAIN is a log phase parameter. If DRAIN is specified when a log phase is not needed, an error message is issued.

**WRITERS**

Specifies that Db2 drains only the writers during the log phase after the MAXRO threshold is reached and then issues DRAIN ALL on entering the switch phase.

**ALL**

Specifies that Db2 drains all readers and writers during the log phase, after the MAXRO threshold is reached.

Consider specifying DRAIN ALL if the following conditions are both true:

- SQL update activity is high during the log phase.
- The default behavior results in a large number of -911 SQL error messages.

**Related information:**

"Claim classes that REORG TABLESPACE drains” on page 618

**LONGLOG**

Specifies the action that Db2 is to perform, after sending a message to the console, if the number of records that the next iteration of logging is to process is not sufficiently lower than the number that the previous iterations processed. This situation means that the reading of the log by the REORG TABLESPACE utility is not being done at the same time as the writing of the application log. LONGLOG is a log phase parameter. If LONGLOG is specified when a log phase is not needed, an error message is issued.

**CONTINUE**

Specifies that until the time on the JOB statement expires, Db2 is to continue performing reorganization, including iterations of log processing, if the estimated time to perform an iteration exceeds the time that is specified for MAXRO.
A value of DEFER for MAXRO and a value of CONTINUE for LONGLOG together mean that REORG is to continue allowing access to the original copy of the area that is being reorganized and does not switch to the shadow copy. The user can execute the ALTER UTILITY command with a large value for MAXRO to initiate switching.

**TERM**

Specifies that Db2 is to terminate the reorganization after the delay that is specified by the DELAY parameter.

**DRAIN**

Specifies that Db2 is to drain the write claim class after the delay that is specified by the DELAY parameter. This action forces the final iteration of log processing to occur. DRAIN is a log phase parameter. If DRAIN is specified when a log phase is not needed, an error message is issued.

**DELAY integer**

Specifies the minimum interval between the time that REORG sends the LONGLOG message to the console and the time that REORG performs the action that is specified by the LONGLOG parameter. DELAY is a log phase parameter. If DELAY is specified when a log phase is not needed, an error message is issued.

`integer` is the number of seconds.

The default value is **1200**.

**TIMEOUT**

Specifies the action that is to be taken if the REORG utility gets a timeout condition while trying to drain an object in either the log or switch phases.

**TERM**

Indicates that Db2 is to behave as follows if you specify the TERM option and a timeout condition occurs:

1. Db2 issues an implicit TERM UTILITY command, causing the utility to end with a return code 8.
2. Db2 issues the DSNU590I and DSNU170I messages.
3. Db2 leaves the object in a read-write state.

**ABEND**

Indicates that, if a timeout condition occurs, Db2 takes one of the following actions:

- If DRAIN ALL is specified, Db2 leaves the object in a UTRW state.
- If DRAIN WRITERS is specified or used by default:
  - If the failure occurs when there is a write drain lock on the object, Db2 leaves the object in a UTRW state.
  - If the failure occurs when there is a read drain lock on the object, Db2 leaves the object in a UTRO state.

**LOGRANGES**

Specifies whether REORG is to use SYSLGRNX information for the LOG phase.

**YES**

REORG uses SYSLGRNX information for the LOG phase whenever possible. This option is the default behavior.

**NO**

REORG does not use SYSLGRNX information for the LOG phase. This option can cause REORG to run much longer. In a data sharing environment this option can result in the merging of all logs from all members. This option is feasible when there is a known integrity issue.
with SYSLGRNX entries and performance problems in accessing
SYSLGRNX for log read determination.

**DRAIN_ALLPARTS**
Specifies the action to take during a part level REORG TABLESPACE
SHRLEVEL REFERENCE or CHANGE when a nonpartitioned secondary index
is defined on a partitioned table space.

- **NO** REORG drains the target data partitions serially followed by the
  nonpartitioned secondary indexes. This option is the default behavior.
- **YES** REORG obtains the table space level drain on the entire partitioned table
  space first, before draining the target data partitions and the indexes. This
  option can provide relief by eliminating drain timeout or deadlocks caused
  by the reverse order of object-draining by REORG and object-claiming by
  DML statements.

**SWITCHTIME**
Specifies the time for the final log iteration of the LOG phase to begin. The
final result and all of the time stamp calculations of SWITCHTIME are in
TIMESTAMP(6). This keyword can be specified with the MAXRO keyword. If
MAXRO DEFER is not specified, REORG enters the final log iteration of the
LOG phase before the specified SWITCHTIME value if the specified or
defaulted MAXRO criteria is met. When MAXRO DEFER is specified, REORG
does not attempt to enter to the final log iteration until the specified
SWITCHTIME is met or affected by an external ALTER UTILITY command in
the changing of its MAXRO value.

- **NONE** Does not specify a time for the final log iteration of the LOG phase. This
option is the default behavior.

  - **timestamp** Specifies the time that the final log iteration of the LOG phase is to begin.
    This time must not have already occurred when REORG is run.

  - **labeled-duration-expression** Calculates the time for the final log iteration of LOG phase is to begin. The
calculation is based on either CURRENT TIMESTAMP or CURRENT
DATE. You can add or subtract one or more constant values to specify the
switch time. This switch time must not have already occurred when
REORG is run. CURRENT TIMESTAMP and CURRENT DATE are
evaluated once, when the REORG statement is first processed. If a list of
objects is specified, the same value is in effect for all objects in the list.

  - **CURRENT_DATE** Specifies that the deadline is to be calculated based on the CURRENT
DATE.

  - **CURRENT_TIMESTAMP** Specifies that the deadline is to be calculated based on the CURRENT
TIMESTAMP.

  - **WITH TIME ZONE** Specifies that the CURRENT_TIMESTAMP is compared with the time
zone column. The time stamp precision of the special register
CURRENT_TIMESTAMP should be the same as the column time stamp
precision. Otherwise, the default time stamp precision is used. The
time zone of CURRENT_TIMESTAMP is the value of special register
CURRENT TIMEZONE. The comparison is done by comparing the Coordinated Universal Time portion of the time stamp.

constant
Indicates a unit of time and is followed by one of the seven duration keywords: YEARS, MONTHS, DAYS, HOURS, MINUTES, SECONDS, or MICROSECONDS. The singular form of these words is also acceptable: YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, MICROSECOND.

NEWMAXRO
Specifies the maximum amount of time for the last log iteration after SWITCHTIME is met. The SWITCHTIME keyword must also be specified. This value overrides the existing MAXRO parameter that is specified. The default is NONE.

NONE
Specifies that when the specified SWITCHTIME is met, REORG proceeds to the last log iteration without taking log processing time in to consideration. Specifying NONE will result in REORG entering the last log iteration almost immediately at or after the specified SWITCHTIME. This option is the default.

integer
integer is the number of seconds. Specifying a small positive value reduces the length of the period of read-only access, but it might increase the elapsed time for REORG to complete. Specifying a large positive value probably ensures that REORG will enter the last log iteration almost immediately at or after the specified SWITCHTIME.

FORCE
Specifies the action to be taken when the utility is draining the table space.

When REORG FORCE is canceling the threads, it performs a soft cancel similar to the cancel that the CANCEL THREAD does.

NONE
Specifies that no action is taken when REORG performs drain. The REORG utility waits for the claimers to commit. The utility will timeout or restart when the drain fails, as determined by existing conditions.

READERS
Specifies that read claimers are canceled when REORG is requesting a drain all on the last RETRY processing.

ALL
Specifies that both read and write claimers are canceled when REORG is requesting a drain all or drain writers on the last RETRY processing.

SORTNPSI
Specifies when REORG TABLESPACE PART is to sort all keys of a non-partitioned secondary index. This keyword is ignored for a REORG that is not partition-level or a REORG without non-partitioned secondary indexes. If SORTNPSI is not specified, the value is determined by REORG_PART_SORT_NPSI subsystem parameter. The benefit of sorting all keys of a non-partitioned secondary index increases as the ratio of data that is reorganized to total data in the table space increases.

The default value is the value of subsystem parameter REORG_PART_SORT_NPSI.
When SORTNPSI AUTO or SORTNPSI YES is specified, or when the REORG_PART_SORT_NPSI subsystem parameter is set to AUTO or YES, REORG TABLESPACE PART can collect statistics for a non-partitioned secondary index when the STATISTICS INDEX keywords are also specified. However, in some of these cases, REORG does not collect statistics. This situation occurs if REORG chooses not to sort all of the nonpartitioned index keys, because the amount of data to reorganize or the sizes of objects exceeded internal thresholds.

**AUTO**

Specifies that if sorting all keys of the non-partitioned secondary indexes improves the elapsed time and CPU performance, all keys are sorted.

**YES**

Specifies that if sorting all keys of the non-partitioned secondary indexes improves the elapsed time, all keys are sorted.

**NO**

Specifies that only keys of the non-partitioned secondary indexes that are in the scope of the REORG are sorted.

**AUX**

Specifies that the LOB table spaces associated with the partitions of a partitioned table space being reorganized by the REORG utility are also reorganized.

**NO**

Indicates that a reorganization is performed on the base table space, but the associated LOB table spaces are not reorganized.

If the AUX keyword is omitted, AUX NO is the default unless one or more of the cases described in AUX YES are true.

AUX NO is ignored when the target table space has pending definition changes to convert it from a simple or segmented table space to a partition-by-growth table space. In this case, AUX YES is in effect.

LOG NO cannot be specified for a REORG operation that completes recovery to a point in time before pending definition changes were materialized, if there were pending definition changes on the base table space and on the LOB table space. REORG must be run on the LOB table space first, and then run on the base table space. When REORG is run on the base table space, AUX YES is in effect.

For a table with LOB columns that are affected by pending alter limit keys, a REORG job with AUX NO does not materialize the limit key changes. In this case, you need to specify AUX YES for those changes to be materialized.

**YES**

Indicates that LOB table spaces associated with the base partitioned table space are reorganized when the base table space is reorganized. Partitions of the associated table spaces are also reorganized.

If the AUX keyword is omitted, in the following cases, AUX YES is the default:

- REORG TABLESPACE of a partition-by-growth base table space with one or more LOB columns, where the table space has a MAXPARTITIONS value that is greater than one.
- REORG TABLESPACE SHRLEVEL REFERENCE REBALANCE of a partitioned base table space with one or more LOB columns.
- REORG TABLESPACE is run against directory table space SPT01, and SPT01 is in the REORP or AREOR state. In this case, AUX YES is always used.

- REORG TABLESPACE of a partitioned base table space with one or more LOB columns where one or more partition ranges are in REORG pending state because an ALTER TABLE PARTITION command has been issued to change the partition key boundaries.

- REORG TABLESPACE DISCARD of a table in a partitioned table space with one or more LOB columns.

When AUX YES is implicitly or explicitly specified, and the COPYDDN parameter specifies a TEMPLATE utility control statement with the &SN, or &TS. variables without substring notation on them, REORG takes the following actions for the LOB table spaces:

- Creates inline image copies
- Resets COPY-pending status

When AUX YES is implicitly or explicitly specified and templates are specified, make sure that those templates generate unique data set names for the auxiliary table spaces that are being reorganized. Make sure that you account for auxiliary table spaces that are included in any specified LISTDEF lists.

When AUX YES is implicitly or explicitly specified, and FlashCopy image copies are taken as part of REORG, REORG produces image copies for all of the LOB table spaces that are being reorganized.

**Restrictions:** When REORG with AUX YES is run on a partition-by-growth table space with LOB columns, the following restrictions apply:

- If REORG generates a new partition during the LOG phase, REORG cannot create inline image copies for LOB table spaces for the newly created partition. REORG leaves the LOB tables space in COPY-pending status and issues a warning message.

- If you specify that REORG is to create inline copies and use a template for the copies, do not use the STACK YES option for the template. If you do so, REORG fails, because the base and auxiliary table spaces cannot be stacked on the same tape volume. If you need to use a template with the STACK YES option, specify AUX NO on the REORG statement and then reorganize and copy the auxiliary table spaces separately.

**Recommendation:** After running REORG with AUX YES on a partition-by-growth table space with LOB columns, run COPY with SCOPE PENDING and a LISTDEF utility control statement that includes the LOB table spaces. Doing so creates a recovery base and removes COPY-pending status for the LOB table spaces.

**FASTSWITCH**

Specifies which switch methodology is to be used for a given reorganization.

When FASTSWITCH is specified with SHRLEVEL CHANGE or SHRLEVEL REFERENCE, the UTILITY OBJECT_CONVERSION subsystem parameter setting NONE, BASIC, or EXTENDED is accepted.

**YES**

Enables the SWITCH phase to use the FASTSWITCH methodology. This option is not allowed for the catalog (DSNDB06) or directory (DSNDB01).

**NO**

Causes the SWITCH phase to use IDCAMS RENAME.
When FASTSWITCH NO is specified with SHRLEVEL REFERENCE or SHRLEVEL CHANGE, pending definition changes are not materialized.

**OFFPOSLIMIT integer**

The OFFPOSLIMIT option is deprecated, and the alternative is running DSNACCOX. Indicates that the specified value is to be compared to the value that Db2 calculates for the explicit clustering indexes of every table in the specified partitions that are in SYSIBM.SYSINDEXPART. The calculation is computed as follows:

\[(\text{NEAROFFPOSF} + \text{FAROFFPOSF}) \times 100 / \text{CARDF}\]

Alternatively, Db2 checks the values in SYSINDEXPART for a single nonpartitioned table space, or for each partition if you specified an entire partitioned table space as the target object. If at least one calculated value exceeds the OFFPOSLIMIT value, REORG is performed or recommended. This option is valid for non-LOB table spaces only.

*integer* is the value that is to be compared and can range from 0 to 65535.

The default value is **10**.

**INDREFLIMIT integer**

The INDREFLIMIT option is deprecated, and the alternative is running DSNACCOX. Indicates that the specified value is to be compared to the value that Db2 calculates for the specified partitions in SYSIBM.SYSTABLEPART for the specified table space. The calculation is computed as follows:

\[(\text{NEARINDREF} + \text{FARINDREF}) \times 100 / \text{CARDF}\]

Alternatively, Db2 checks the values in SYSTABLEPART for a single nonpartitioned table space, or for each partition if you specified an entire partitioned table space as the target object. If at least one calculated value exceeds the INDREFLIMIT value, REORG is performed or recommended. This option is valid for non-LOB table spaces only.

*integer* is the value that is to be compared and can range from 0 to 65535.

The default value is **10**.

**REPORTONLY**

The REPORTONLY option is deprecated, and the alternative is running DSNACCOX. Specifies that REORG is only to be recommended, not performed. REORG produces a report with one of the following return codes:

1. No limit met; no REORG is to be performed or recommended.
2. REORG is to be performed or recommended.

**UNLOAD**

Specifies whether the utility job is to continue processing or end after the data is unloaded. Unless you specify UNLOAD EXTERNAL, data can be reloaded only into the same table and table space (as defined in the Db2 catalog) on the same subsystem. (This does not preclude VSAM redefinition during UNLOAD PAUSE.)

You must specify UNLOAD ONLY for the data set to be in a format that is compatible with the FORMAT UNLOAD option of LOAD. However, with LOAD, you can load the data only into the same object from which it is unloaded.

This option is valid for non-LOB table spaces only.
You must specify UNLOAD EXTERNAL for the data set to be in a format that is usable by LOAD without the FORMAT UNLOAD option. With UNLOAD EXTERNAL, you can load the data into any table with compatible columns in any table space on any Db2 subsystem.

**CONTINUE**

Specifies that, after the data has been unloaded, the utility is to continue processing. An edit routine can be called to decode a previously encoded data row if an index key requires extraction from that row.

If you specify DISCARD, rows are decompressed and edit routines are decoded. If you also specify DISCARD to a file, rows are decoded by field procedure, and the following columns are converted to Db2 external format:

- SMALLINT
- INTEGER
- FLOAT
- DECIMAL
- TIME
- TIMESTAMP

Otherwise, edit routines or field procedures are bypassed on both the UNLOAD and RELOAD phases for table spaces. Validation procedures are not invoked during either phase.

**PAUSE**

The UNLOAD PAUSE option is deprecated, and the alternative is running the UNLOAD utility. Specifies that, after the data has been unloaded, processing is to end. The utility stops and the RELOAD status is stored in SYSIBM.SYSUTIL so that processing can be restarted with RELOAD RESTART(PHASE).

This option is useful if you want to redefine data sets during reorganization. For example, with a user-defined data set, you can:

- Run REORG with the UNLOAD PAUSE option.
- Redefine the data set by using Access Method Services.
- Restart REORG by resubmitting the previous job and specifying RESTART(PHASE).

However, you cannot use UNLOAD PAUSE if you specify the LIST option.

**ONLY**

The UNLOAD ONLY option is deprecated, and the alternative is running the UNLOAD utility. Specifies that, after the data has been unloaded, the utility job ends and the status that corresponds to this utility ID is removed from SYSIBM.SYSUTIL.

If you specify UNLOAD ONLY with REORG TABLESPACE, any edit routine or field procedure is executed during record retrieval in the unload phase.

This option is not allowed for any table space in DSNDB01 or DSNDB06. The DISCARD and WHEN options are not allowed with UNLOAD ONLY.

**EXTERNAL**
The UNLOAD EXTERNAL option is deprecated, and the alternative is running the UNLOAD utility. Specifies that, after the data has been unloaded, the utility job is to end and the status that corresponds to this utility ID is removed.

The UNLOAD utility has more functions. If you specify UNLOAD EXTERNAL with REORG TABLESPACE, rows are decompressed, edit routines are decoded, field procedures are decoded, and SMALLINT, INTEGER, FLOAT, DECIMAL, DATE, TIME, and TIMESTAMP columns are converted to Db2 external format. Validation procedures are not invoked.

Do not specify the EXTERNAL keyword for:
- Table spaces in DSNDB01 or DSNDB06
- Base tables with XML columns
- XML table spaces

The DISCARD option is not allowed with UNLOAD EXTERNAL.

**NOPAD**

Specifies whether the variable-length columns in the unloaded or discarded records are to occupy the actual data length without additional padding. The unloaded records can have varying lengths.

**YES**

Specifies that the variable-length columns are to have no additional padding.

**NO**

Specifies that REORG processing pads variable-length columns in the unloaded or discarded records to their maximum length; the unloaded or discarded records have equal lengths for each table.

You can specify the NOPAD option only with UNLOAD EXTERNAL or with UNLOAD DISCARD.

Although the LOAD utility processes records with variable-length columns that were unloaded or discarded with the NOPAD option, these records cannot be processed by applications that process only fields that are in fixed positions.

For the generated LOAD statement to provide a NULLIF condition for fields that are not in a fixed position, Db2 generates an input field definition with a name in the form of DSN_NULL_IND_nnnn, where nnnn is the number of the associated column.

For example, the LOAD statement that is generated for the EMP sample table looks similar to the LOAD statement that is in the following figure:

*Figure 60. Sample LOAD statement generated by REORG TABLESPACE with the NOPAD keyword*

```
LOAD DATA INDDN SYSREC LOG NO RESUME YES
EBCDIC CCSID(00500,00000,00000)
INTO TABLE "DSN8B10"."EMP"
WHEN(00004:00005 = X'0012')
("EMPN"
  "FIRSTNAME"  "POSITION(+*)"  "CHAR(006)
  "MIDINIT"  "POSITION(+*)"  "CHAR(001)
  "LASTNAME"  "POSITION(+*)"  "VARC".
  DSN_NULL_IND_00005  "POSITION(+*)"  "CHAR(003)
  "WORKDEPT"  "POSITION(+*)"  "CHAR(003)
  NULLIF(DSN_NULL_IND_00005)=X'FF'
```
FROM TABLE

Specifies the tables that are to be reorganized. The table space that is specified in REORG TABLESPACE can store more than one table. All tables that are specified by FROM TABLE statements must be unique. All tables are unloaded for UNLOAD EXTERNAL, and all tables might be subject to DISCARD. If you specify UNLOAD EXTERNAL and want to limit which tables and rows are unloaded, specify FROM TABLE with the WHEN option. If you specify DISCARD, you must qualify the rows that you want to discard by specifying FROM TABLE with the WHEN option.

Do not specify FROM TABLE with REORG TABLESPACE LIST.

table-name

Specifies the name of the table that is to be qualified by the following WHEN clause. The table must be described in the catalog and must not be a catalog table. If the table name is not qualified by a schema name, the authorization ID of the person who invokes the utility job step is used as the schema qualifier of the table name. Enclose the table name in quotation marks if the name contains a blank.

WHEN

Indicates which records in the table space are to be unloaded (for UNLOAD EXTERNAL) or discarded (for DISCARD). If you do not specify a WHEN clause for a table in the table space, all of the records are unloaded (for UNLOAD EXTERNAL), or none of the records is discarded (for DISCARD).

The option following WHEN describes the conditions for UNLOAD or DISCARD of records from a table and must be enclosed in parentheses.

selection condition

Specifies a condition that is true, false, or unknown about a specific row. When the condition is true, the row qualifies for UNLOAD or DISCARD. When the condition is false or unknown, the row does not qualify.
A selection condition consists of at least one predicate and any logical operators (AND, OR, NOT). The result of a selection condition is derived by applying the specified logical operators to the result of each specified predicate. If logical operators are not specified, the result of the selection condition is the result of the specified predicate.

Selection conditions within parentheses are evaluated first. If the order of evaluation is not specified by parentheses, AND is applied before OR.

If the control statement is in the same encoding scheme as the input data, you can code character constants in the control statement. Otherwise, if the control statement is not in the same encoding scheme as the input data, you must code the condition with hexadecimal constants.

If the target table is ASCII, any character constants must be specified in hexadecimal. For example, if the table space is in EBCDIC and the control statement is in UTF-8, use (1:1)=X'F1' in the condition rather than (1:1)='1'.

**Restriction:** REORG TABLESPACE cannot filter rows based on column-level encrypted data.

**predicate**
A predicate specifies a condition that is true, false, or unknown about a given row or group.

**basic predicate**
Specifies the comparison of a column with a constant. If the value of the column is null, the result of the predicate is unknown. Otherwise, the result of the predicate is true or false.

**Predicate**
*Is true if and only if*

**column-name**\(=\) **constant**
The column is equal to the constant or labeled duration expression.

**column-name**\(\ne\) **constant**
The column is not equal to the constant or labeled duration expression.

**column-name**\(>\) **constant**
The column is greater than the constant or labeled duration expression.

**column-name**\(<\) **constant**
The column is less than the constant or labeled duration expression.

**column-name**\(\ge\) **constant**
The column is greater than or equal to the constant or labeled duration expression.

**column-name**\(\le\) **constant**
The column is less than or equal to the constant or labeled duration expression.

**Comparison operators:** The following forms of the comparison operators are also supported in basic and quantified predicates: !\(=\), !\(<\), and !\(>\), where ! means not. In addition, in code pages 437, 819, and 850, the forms \(\sim=\), \(\sim<\), and \(\sim>\) are supported. All these product-specific forms of the comparison operators are intended only to support
existing REORG statements that use these operators and are not recommended for use in new REORG statements.

A not sign (¬), or the character that must be used in its place in certain countries, can cause parsing errors in statements that are passed from one DBMS to another. The problem occurs if the statement undergoes character conversion with certain combinations of source and target CCSIDs. To avoid this problem, substitute an equivalent operator for any operator that includes a not sign. For example, substitute ‘< >’ for ‘¬=’, ‘<=’ for ‘¬>’, and ‘>=’ for ‘¬<’.

**BETWEEN predicate**

Indicates whether a given value is between two other given values that are specified in ascending order. Each of the predicate’s two forms (BETWEEN and NOT BETWEEN) has an equivalent search condition, as shown in the following table. If relevant, the table also shows any equivalent predicates.

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Equivalent predicate</th>
<th>Equivalent search condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>column BETWEEN value1 AND value2</td>
<td>None</td>
<td>(column &gt;= value1 AND column &lt;= value2)</td>
</tr>
<tr>
<td>column NOT BETWEEN value1 AND value2</td>
<td>NOT(column BETWEEN value1 AND value2)</td>
<td>(column &lt; value1 OR column &gt; value2)</td>
</tr>
</tbody>
</table>

**Note:** The values can be constants or labeled duration expressions.

For example, the following predicate is true for any row when salary is greater than or equal to 10 000 and less than or equal to 20 000:

SALARY BETWEEN 10000 AND 20000

**labeled-duration-expression**

Specifies an expression that begins with the following special register values:

- CURRENT DATE (CURRENT_DATE is acceptable.)
- CURRENT TIMESTAMP (CURRENT_TIMESTAMP is acceptable.)

Optionally, the expression contains the arithmetic operations of addition or subtraction, expressed by a number followed by one of the seven duration keywords:

- YEARS (or YEAR)
- MONTHS (or MONTH)
- DAYS (or DAY)
- HOURS (or HOUR)
- MINUTES (or MINUTE)
- SECONDS (or SECOND)
- MICROSECONDS (or MICROSECOND)

Utilities evaluate a labeled-duration-expression as a timestamp and implicitly perform a conversion to a date if the comparison is with a date column.

**Incrementing and decrementing CURRENT DATE:** The result of adding a duration to a date, or of subtracting a duration from a date, is itself a date. (For the purposes of this operation, a month denotes the
equivalent of a calendar page. Adding months to a date, then, is like turning the pages of a calendar, starting with the page on which the date appears.) The result must fall between the dates January 1, 0001 and December 31, 9999 inclusive.

The following table describes the effects of adding and subtracting years, months, days, and other dates.

<table>
<thead>
<tr>
<th>Value that is added or subtracted</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Years</td>
<td>Adding or subtracting a duration of years affects only the year portion of the date. The month is unchanged, as is the day unless the result would be February 29 of a non-leap-year. In this case, the day portion of the result is set to 28.</td>
</tr>
<tr>
<td>Months</td>
<td>Adding or subtracting a duration of months affects only months and, if necessary, years. The day portion of the date is unchanged unless that day does not exist in the resulting month. (September 31, for example). In this case the day is set to the last day of the month.</td>
</tr>
</tbody>
</table>

Adding a month to a date gives the same day one month later unless that day does not exist in the later month. In that case, the day in the result is set to the last day of the later month. For example, January 28 plus one month gives February 28; one month added to January 29, 30, or 31 results in either February 28 or, for a leap year, February 29. If one or more months is added to a given date and then the same number of months is subtracted from the result, the final date is not necessarily the same as the original date.

<table>
<thead>
<tr>
<th>Days</th>
<th>Adding or subtracting a duration of days affects the day portion of the date, and potentially the month and year.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dates</td>
<td>When a positive date duration is added to a date, or a negative date duration is subtracted from a date, the date is incremented by the specified number of years, months, and days.</td>
</tr>
</tbody>
</table>

When a positive date duration is subtracted from a date, or a negative date duration is added to a date, the date is decremented by the specified number of days, months, and years.

The order in which labeled date durations are added to and subtracted from dates can affect the results. When you add labeled date durations to a date, specify them in the order of YEARS + MONTHS + DAYS. When you subtract labeled date durations from a date, specify them in the order of DAYS - MONTHS - YEARS. For example, to add one year and one day to a date, specify the following code:

CURRENT DATE + 1 YEAR + 1 DAY

To subtract one year, one month, and one day from a date, specify the following code:

CURRENT DATE - 1 DAY - 1 MONTH - 1 YEAR

**Incrementing and decrementing timestamps:** The result of adding a duration to a timestamp, or of subtracting a duration from a timestamp, is itself a timestamp. Date and time arithmetic is performed as previously defined, except that an overflow or underflow of hours is carried into the date part of the result, which must be within the range
of valid dates. For example, if the current date is January 15 and the current time is 20:00, \texttt{CURRENT\_TIMESTAMP+8\ HOURS} yields January 16, 04:00. Likewise, \texttt{CURRENT\_TIMESTAMP-22\ HOURS} yields January 14, 22:00.

**IN predicate**

Specifies that a value is to be compared with a set of values. In the IN predicate, the second operand is a set of one or more values that are specified by constants. Each of the predicate's two forms (IN and NOT IN) has an equivalent search condition, as shown in the following table.

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Equivalent search condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{value1 IN\ (value1, value2,..., valuen)}</td>
<td>(value1 = value2 OR ... OR value1 = valuen)</td>
</tr>
<tr>
<td>\texttt{value1 NOT IN\ (value1, value2,..., valuen)}</td>
<td>value1 ≠ value2 AND ... AND value1 ≠ valuen</td>
</tr>
</tbody>
</table>

**Note:** The values can be constants or labeled duration expressions.

For example, the following predicate is true for any row with an employee in department D11, B01, or C01:

\texttt{WORKDEPT IN ('D11', 'B01', 'C01')}

**LIKE predicate**

Qualifies strings that have a certain pattern. Specify the pattern by using a string in which the underscore and percent sign characters can be used as wildcard characters. The underscore character (_) represents a single, arbitrary character. The percent sign (%) represents a string of zero or more arbitrary characters.

In this description, let \texttt{x} denote the column that is to be tested and \texttt{y} denote the pattern in the string constant.

The following rules apply to predicates of the form “\texttt{x LIKE y}...”. If NOT is specified, the result is reversed.

- When \texttt{x} or \texttt{y} is null, the result of the predicate is unknown.
- When \texttt{y} is empty and \texttt{x} is not empty, the result of the predicate is false.
- When \texttt{x} is empty and \texttt{y} is not empty, the result of the predicate is false unless \texttt{y} consists only of one or more percent signs.
- When \texttt{x} and \texttt{y} are both empty, the result of the predicate is true.
- When \texttt{x} and \texttt{y} are both not null, the result of the predicate is true if \texttt{x} matches the pattern in \texttt{y} and false if \texttt{x} does not match the pattern in \texttt{y}.

The pattern string and the string that is to be tested must be of the same type; that is, both \texttt{x} and \texttt{y} must be character strings, or both \texttt{x} and \texttt{y} must be graphic strings. When \texttt{x} and \texttt{y} are graphic strings, a character is a DBCS character. When \texttt{x} and \texttt{y} are character strings and \texttt{x} is not mixed data, a character is an SBCS character, and \texttt{y} is interpreted as SBCS data regardless of its subtype.

Within the pattern, a percent sign (%) or underscore character (_) can represent the literal occurrence of a percent sign or underscore character. To have a literal meaning, each character must be preceded by an escape character.

The ESCAPE clause designates a single character. You can use that character, and only that character, multiple times within the pattern as
an escape character. When the ESCAPE clause is omitted, no character serves as an escape character and percent signs and underscores in the pattern can only be used to represent arbitrary characters; they cannot represent their literal occurrences.

The following rules apply to the use of the ESCAPE clause:

- The ESCAPE clause cannot be used if \( x \) is mixed data.
- If \( x \) is a character string, the data type of the string constant must be character string. If \( x \) is a graphic string, the data type of the string constant must be graphic string. In both cases, the length of the string constant must be 1.
- The pattern must not contain the escape character except when followed by the escape character, '%', or '_'. For example, if '+' is the escape character, any occurrences of '+' other than '++', '+_', or '+%' in the pattern is an error.

When that pattern does not include escape characters, a simple description of its meaning is:

- The underscore character (_) represents a single, arbitrary character.
- The percent sign (%) represents a string of zero or more arbitrary characters.
- Any other character represents a single occurrence of itself.

**Strings and patterns:**

The string \( y \) is interpreted as a sequence of the minimum number of substring specifiers, such that each character of \( y \) is part of exactly one substring specifier. A substring specifier is an underscore, a percent sign, or any non-empty sequence of characters other than an underscore or percent sign.

The string \( x \) matches the pattern \( y \) if a partitioning of \( x \) into substrings exists, such that:

- A substring of \( x \) is a sequence of zero or more contiguous characters, and each character of \( x \) is part of exactly one substring.
- If the \( n \)th substring specifier is an underscore, the \( n \)th substring of \( x \) is any single character.
- If the \( n \)th substring specifier is a percent sign, the \( n \)th substring of \( x \) is any sequence of zero or more characters.
- If the \( n \)th substring specifier is neither an underscore nor a percent sign, the \( n \)th substring of \( x \) is equal to that substring specifier and has the same length as that substring specifier.
- The number of substrings of \( x \) is the same as the number of substring specifiers.

When escape characters are present in the pattern string, an underscore, percent sign, or escape character represents a single occurrence of itself if and only if it is preceded by an odd number of successive escape characters.

The way a pattern is matched to evaluate the LIKE predicate depends on whether blanks at the end of fixed length strings are significant, or
if the blanks are ignored. When the LIKE_BLANK_INSIGNIFICANT subsystem parameter is enabled, the LIKE predicate can produce different results.

**Mixed-data patterns:**

If $x$ is mixed data, the pattern is assumed to be mixed data, and its special characters are interpreted as follows:

- A single-byte underscore refers to one single-byte character; a double-byte underscore refers to one double-byte character.
- A percent sign, either single-byte or double-byte, refers to any number of characters of any type, either single-byte or double-byte.
- Redundant shift bytes in $x$ or $y$ are ignored.

**Related information:**

[LIKE predicate (Db2 SQL)]

**NULL predicate**

Specifies a test for null values.

If the value of the column is null, the result is true. If the value is not null, the result is false. If NOT is specified, the result is reversed.

**KEEPDICTIONARY**

Prevents REORG TABLESPACE from building a new compression dictionary when unloading the rows. The REORG utility builds the compression dictionary during the UNLOAD process. This dictionary is then used during the RELOAD phase to compress the data.

The efficiency of REORG increases with the KEEPDICTIONARY option for the following reasons:

- The processing cost of building the compression dictionary is eliminated.
- Existing compressed rows do not need to be compressed again.
- Existing compressed rows do not need to be expanded, unless indexes require it or SORTDATA is used.

Possible reasons for not specifying KEEPDICTIONARY are:

- If the data has changed significantly since the last dictionary was built, rebuilding the dictionary might save a significant amount of space.
- If the current dictionary was built either by the LOAD utility or automatically by Db2 based on records that have been inserted over time, rebuilding the dictionary by using REORG might produce a better compression dictionary.
- If the data is being converted from basic row format to reordered row format, REORG builds a new dictionary for the new format. Db2 ignores the KEEPDICTIONARY option if the REORG utility changes the table space from basic row format to reordered row format.
- If REORG is materializing a pending alter of the buffer pool, REORG builds a new dictionary. Db2 ignores the KEEPDICTIONARY option if REORG is materializing a buffer pool change.

KEEPDICTIONARY is valid only if a compression dictionary exists and the table space or partition that is being reorganized has the COMPRESS YES attribute. If a dictionary does not exist, one is built, a warning message is issued, and all the records are compressed.
Messages DSNU234I and DSNU244I, which show compression statistics, are not issued when you specify REORG UNLOAD CONTINUE KEEPDICTIONARY or REORG UNLOAD PAUSE KEEPDICTIONARY.

REORG ignores the KEEPDICTIONARY option if a partition that is being reorganized is in REORG-pending status.

Note: You must use KEEPDICTIONARY to ensure that the compression dictionary is maintained.

Related information:
“Compressing data by using the LOAD utility” on page 330

STATISTICS
Specifies that statistics for the table space or associated index, or both, are to be gathered; the statistics are reported or stored in the Db2 catalog. If statistics are collected with the default options, only the statistics for the table space are updated.

If you specify a table space partition or a range of partitions along with the STATISTICS keyword, Db2 collects statistics only for the specified table space partitions. This option is valid for non-LOB table spaces only.

If you specify a base table space with the STATISTICS keyword, Db2 does not gather statistics for the related XML table space or its indexes.

When SORTNPSI AUTO or SORTNPSI YES is specified, or when the REORG_PART_SORT_NPSI subsystem parameter is set to AUTO or YES, REORG TABLESPACE PART can collect statistics for a non-partitioned secondary index when the STATISTICS INDEX keywords are also specified. However, in some of these cases, REORG does not collect statistics. This situation occurs if REORG chooses not to sort all of the nonpartitioned index keys, because the amount of data to reorganize or the sizes of objects exceeded internal thresholds.

Restrictions:
• If you specify STATISTICS for encrypted data, Db2 might not provide useful statistics on this data.
• You cannot specify STATISTICS if you specify the CLONE keyword.

Statistics for both a table space and its associated indexes are collected and updated in the Db2 catalog when pending definition changes are materialized during REORG TABLESPACE with SHRLEVEL REFERENCE or CHANGE.

If the STATISTICS keyword is specified in the REORG TABLESPACE statement, the options specified overwrite the default options.

If the STATISTICS keyword is not specified in the REORG TABLESPACE statement, the following keywords are used by default:
• STATISTICS TABLE ALL
• INDEX ALL
• UPDATE ALL
• HISTORY ALL

Recommendation: Partition statistics can become obsolete. The partition statistics that can be obsolete are COLGROUP statistics, statistics for key column values in indexes, HISTOGRAM statistics, frequency statistics with
NUMCOLS > 1, and statistics for extended indexes where applicable. Run the RUNSTATS utility to collect the partition statistics again.

All tables identified by the STATISTICS TABLE keyword must belong to the table space that is specified in the TABLESPACE option.

TABLE
Specifies the table for which column information is to be gathered.

Do not specify STATISTICS TABLE table-name with the LIST keyword. Instead, specify STATISTICS TABLE (ALL).

(ALL)
Specifies that information is to be gathered for all columns of all tables in the table space.

(table-name)
Specifies the tables for which column information is to be gathered. If you omit the qualifier, the user identifier for the utility job is used. Enclose the table name in quotation marks if the name contains a blank.

If you specify more than one table, you must repeat the TABLE option. Multiple TABLE options must be specified entirely before or after any INDEX keyword that may also be specified. For example, the INDEX keyword may not be specified between any two TABLE keywords.

SAMPLE integer
Indicates the percentage of rows to be sampled when collecting statistics on non-leading-indexed columns of an index or non-indexed columns. You can specify any value from 1 through 100.

The default value is 25. The SAMPLE option is not allowed for LOB table spaces.

COLUMN
Specifies columns for which column information is to be gathered.

You can specify this option only if you specify a particular table for which statistics are to be gathered (TABLE (table-name)). If you specify particular tables and do not specify the COLUMN option, the default, COLUMN(ALL), is used. If you do not specify a particular table when using the TABLE option, you cannot specify the COLUMN option; however, COLUMN(ALL) is assumed.

(ALL)
Specifies that statistics are to be gathered for all columns in the table.

(column-name, ...)
Specifies the columns for which statistics are to be gathered.

You can specify a list of column names; the maximum is 10. If you specify more than one column, separate each name with a comma.

INDEX
Specifies indexes for which information is to be gathered. Column information is gathered for the first column of the index. All the indexes must be associated with the same table space, which must be the table space that is specified in the TABLESPACE option.

Do not specify STATISTICS INDEX index-name with the LIST keyword. Instead, specify STATISTICS INDEX (ALL).
(ALL)
   Specifies that the column information is to be gathered for all indexes that
   are defined on tables that are contained in the table space.

(index-name)
   Specifies the indexes for which information is to be gathered. Enclose the
   index name in quotation marks if the name contains a blank.

COLGROUP (column-name, ...)
   Indicates that the specified set of columns are treated as a group. This option
   enables inline statistics to collect a cardinality value on the specified column
   group. Inline statistics ignores COLGROUP when processing XML table spaces
   and indexes.

   When you specify the COLGROUP keyword, inline statistics collects
   correlation statistics for the specified column group. If you want inline
   statistics to also collect distribution statistics, specify the FREQVAL option with
   COLGROUP.

   (column-name, ...) specifies the names of the columns that are part of the
   column group.

   When you define a column group on a single column, you can potentially
   improve RUNSTATS performance by specifying the STATCLGMEMSR parameter
   or changing the value of the STATCLGSRT subsystem parameter. You can use
   these options to avoid column group sorts by external sort programs.

   To specify more than one column group, repeat the COLGROUP option.

Restriction: The length of the COLGROUP value cannot exceed the maximum
   length of the COLVALUE column in the SYSIBM.SYSCOLDIST catalog table.

Related information:

   STATISTICS COLGROUP DATA SORT STG LIMIT field (STATCLGSRT
   subsystem parameter) (Db2 Installation and Migration)

FREQVAL
   Indicates, when specified with the COLGROUP option, that frequency statistics
   are also to be gathered for the specified group of columns. (COLGROUP
   indicates that cardinality statistics are gathered.) One group of statistics is
   gathered for each column. You must specify COUNT integer with COLGROUP
   FREQVAL. The utility ignores FREQVAL MOST/LEAST/BOTH when
   processing XML table spaces and indexes.

COUNT integer
   Indicates the number of frequently occurring values to be collected from
   the specified column group. For example, COUNT 20 means that Db2
   collects 20 frequently occurring values from the column group. When the
   COUNT keyword is not specified, the default is COUNT 10. Specifying a
   value of 1000 or more can increase the prepare time for some SQL
   statements.

   MOST
   Indicates that the utility is to collect the most frequently occurring
   values for the specified set of columns when COLGROUP is specified.

   BOTH
   Indicates that the utility is to collect the most and the least frequently
   occurring values for the specified set of columns when COLGROUP is
   specified.
LEAST
Indicates that the utility is to collect the least frequently occurring
values for the specified set of columns when COLGROUP is specified.

HISTOGRAM
Indicates, when specified with the COLGROUP option, that histogram statistics
are to be gathered for the specified group of columns. Inline statistics ignore
HISTOGRAM when processing XML table spaces and indexes.

Histogram statistics that you collect through inline statistics are not the same
as histogram statistics that you collect through RUNSTATS. Histogram statistics
that you collect with inline statistics are only rough estimates. To obtain more
exact statistics, use RUNSTATS.

NUMQUANTILES  integer
Indicates how many quantiles that the utility collects. The integer value must
be greater than or equal to one. The number of quantiles that you specify must
never exceed the total number of distinct values in the column or the column
group. The maximum number of quantiles is 100.

When the NUMQUANTILES keyword is omitted, NUMQUANTILES takes a
default value of 100. Based on the number of records in the table, the number
of quantiles is readjusted down to an optimal number.

KEYCARD
The KEYCARD option is deprecated in the utility control statement and no
longer needs to be specified to collect cardinality statistics on the values in the
key columns of an index.

When the STATISTICS and INDEX options are specified, the utility always
collects all of the distinct values in all of the 1 to n key column combinations in
an index. n is the number of columns in the index. With the deprecation of
KEYCARD, this functionality cannot be disabled.

The utility tolerates the specification of the KEYCARD option. The utility does
not issue any messages if the control statement includes or excludes the
KEYCARD option when STATISTICS and INDEX are specified.

FREQVAL
Controls the collection of frequent-value statistics. If you specify FREQVAL, it
must be followed by the NUMCOLS keyword.

NUMCOLS
Indicates the number of key columns that are to be concatenated together
when collecting frequent values from the specified index. Specifying '3'
means that frequent values are to be collected on the concatenation of the
first three key columns. The default value is 1, which means that Db2
collects frequent values on the first key column of the index.

COUNT
Indicates the number of frequent values that are to be collected. Specifying
'15' means that Db2 collects 15 frequent values from the specified key
columns. The default value is 10.

HISTOGRAM
Indicates that histogram statistics are requested for the specified index.

NUMCOLS
The number of key columns that are to be concatenated when collecting
histogram statistics from the specified index.
NUMQUANTILES
The integer values that follows NUMQUANTILES indicates the number
quantiles are requested. The integer value must be greater than or equal to
1.

Histogram statistics can be collected only on keys with the same order if the
specified key columns for histogram statistics are of mixed order, a DSNU633I
warning message is issued.

Related information:

REPORT
Specifies whether a set of messages is to be generated to report the collected
statistics.

NO Indicates that the set of messages is not to be sent as output to SYSPRINT.

YES Indicates that the set of messages is to be sent as output to SYSPRINT. The
generated messages are dependent on the combination of keywords (such
as TABLESPACE, INDEX, TABLE, and COLUMN) that are specified with
the RUNSTATS utility. However, these messages are not dependent on the
specification of the UPDATE option. REPORT YES always generates a
report of SPACE and ACCESSPATH statistics.

UPDATE
Indicates whether the collected statistics are to be inserted into the catalog
tables. UPDATE also allows you to select statistics that are used for access path
selection or statistics that are used by database administrators.

ALL Indicates that all collected statistics are to be updated in the catalog.

ACCESSPATH Indicates that only the catalog table columns that provide statistics that are
used for access path selection are to be updated.

SPACE Indicates that only the catalog table columns that provide statistics to help
database administrators assess the status of a particular table space or
index are to be updated.

NONE Indicates that no catalog tables are to be updated with the collected
statistics. This option is valid only when REPORT YES is specified.

STATCLGMEMSRT integer
Specifies the amount of memory that the utility can use for sorting records
when collecting statistics on a single column that is defined with the
COLGROUP option. Use STATCLGMEMSRT to avoid column group sorts by
an external sort program, which can negatively affect the performance of
statistics collection.

integer specifies the number of megabytes of memory space that the utility can
use for an in-memory sort. Valid values are 0 to 4096. If the amount of space
that is needed for the sort exceeds the integer value, the utility invokes a sort
program. If you specify 0, the utility automatically invokes a sort program.
The amount of space that is needed for the column group sort depends on the following factors:

- The number of column groups for which the utility is collecting statistics
- The length of the single-column column group
- The number of distinct values in the column (cardinality)

The value of STATCLGMEMSRT overrides the value of the STATCLGSRT subsystem parameter.

**Related information:**

[STATISTICS COLGROUP DATA SORT STG LIMIT field (STATCLGSRT subsystem parameter) (Db2 Installation and Migration)]

**HISTORY**

Specifies that all catalog table inserts or updates to the catalog history tables are to be recorded.

The default value is whatever value is specified in the STATISTICS HISTORY field on panel DSNTIP6.

**ALL**

Indicates that all collected statistics are to be updated in the catalog history tables.

**ACCESSPATH**

Indicates that only the catalog history table columns that provide statistics that are used for access path selection are to be updated.

**SPACE**

Indicates that only space-related catalog statistics are to be updated in catalog history tables.

**NONE**

Indicates that no catalog history tables are to be updated with the collected statistics.

**FORCEROLLUP**

Specifies whether aggregation or rollup of statistics is to take place when RUNSTATS is executed even if statistics have not been gathered on some partitions; for example, partitions have not had any data loaded. Aggregate statistics are used by the optimizer to select the best access path.

**YES**

Indicates that forced aggregation or rollup processing is to be done, even though some partitions might not contain data.

**NO**

Indicates that aggregation or rollup is to be done only if data is available for all partitions.

If data is not available for all partitions, DSNU623I message is issued if the installation value for STATISTICS ROLLUP on panel DSNTIP6 is set to NO.

**PUNCHDDN**

Specifies the DD statement for a data set that is to receive the LOAD utility control statements that are generated by REORG TABLESPACE UNLOAD EXTERNAL or REORG TABLESPACE DISCARD FROM TABLE ... WHEN.

`ddname` is the DD name.

The default value is **SYSPUNCH**.
PUNCHDDN is required if the limit key of the last partition of a partitioned table space has been reduced.

PUNCHDDN is not valid for LOB table spaces.

The PUNCHDDN keyword specifies either a DD name or a TEMPLATE name specification from a previous TEMPLATE control statement. If utility processing detects that the specified name is both a DD name in the current job step and a TEMPLATE name, the utility uses the DD name.

**DISCARDDN** *ddname*

Specifies the DD statement for a discard data set, which contains copies of records that meet the DISCARD FROM TABLE ... WHEN specification.

*ddname* is the DD name.

If you omit the DISCARDDN option, the utility saves discarded records only if a SYSDISC DD statement is in the JCL input.

The default value is **SYSDISC**.

The DISCARDDN keyword specifies either a DD name or a TEMPLATE name specification from a previous TEMPLATE control statement. If utility processing detects that the specified name is both a DD name in the current job step and a TEMPLATE name, the utility uses the DD name.

**UNLDDN** *ddname*

Specifies the name of the unload data set.

*ddname* is the DD name of the unload data set.

The default value is **SYSREC**.

The UNLDDN keyword specifies either a DD name or a TEMPLATE name specification from a previous TEMPLATE control statement. If utility processing detects that the specified name is both a DD name in the current job step and a TEMPLATE name, the utility uses the DD name.

**SORTDEVT** *device-type*

Specifies the device type for temporary data sets that are to be dynamically allocated by the external sort program.

*device-type* is the device type. You can specify any disk device that is acceptable to the DYNALLOC parameter of the SORT or OPTION control statement for the sort program. Tape devices are not supported by the sort program.

If you omit SORTDEVT and require a sort of the index keys, you must provide the DD statements that the sort program needs for the temporary data sets.

SORTDEVT is ignored for the catalog and directory table spaces that are listed in "Reorganizing the catalog and directory" on page 628.

SORTDEVT cannot be used for LOB table spaces.

The utility does not allow a TEMPLATE specification to dynamically allocate sort work data sets. The SORTDEVT keyword controls dynamic allocation of these data sets.

**SORTNUM** *integer*

Specifies the number of temporary data sets that are to be dynamically allocated for all sorts that REORG performs.

*integer* is the number of temporary data sets that can range from 2 to 255.
If you omit SORTDEVT, SORTNUM is ignored. If you use SORTDEVT and omit SORTNUM, no value is passed to the sort program. The sort program uses its own SORTNUM default value.

You need at least two sort work data sets for each sort. The SORTNUM value applies to each sort invocation in the utility. For example, a total of 24 sort work data sets would be allocated for a job, if the following criteria is true:

- There are three indexes.
- There are no constraints limiting parallelism.
- SORTNUM is specified as 8.

Each sort work data set consumes both above the line and below the line virtual storage. Therefore, if you specify a value for SORTNUM that is too high, the utility might decrease the degree of parallelism due to virtual storage constraints, and possibly decrease the degree down to one, which would mean no parallelism.

Important: The SORTNUM keyword is ignored if the UTSORTAL subsystem parameter is set to YES and the IGNSORTN subsystem parameter is set to YES.

SORTNUM is ignored for the catalog and directory table spaces listed in “Reorganizing the catalog and directory” on page 628.

PREFORMAT
Specifies that the remaining pages are to be preformatted up to the high-allocated RBA in the table space and index spaces that are associated with the table space or partitions that are being reorganized. The preformatting occurs after the data is loaded and the indexes are built.

PREFORMAT can operate on an entire table space and its index spaces, or on a partition of a partitioned table space and its corresponding partitioning index space. When AUX YES is specified or accepted as the default, the LOB table spaces and auxiliary indexes that are associated with the base partitions that are being reorganized are also preformatted at the end of the RELOAD phase.

PREFORMAT is ignored if you specify UNLOAD ONLY or UNLOAD EXTERNAL.

ROWFORMAT
Specifies the output row format in the affected table space or partition. This keyword overrides the existing RRF subsystem parameter setting when the keyword is specified. This keyword has no effect on LOB, catalog, directory, XML table spaces, and Universal table spaces that are participating in a CLONE relationship.

BRF
- Specifies that the table space or partition that is being reorganized or replaced are to be converted to or remain in basic row format.

RRF
- Specifies that the table space or partition that is being reorganized or replaced are to be converted to or remain in reordered row format.

RBALRSN_CONVERSION
Specifies the RBA or LRSN format of the target object after the completion of the REORG utility. If the keyword is not specified, the conversion specified in the UTILITY_OBJECT_CONVERSION subsystem parameter is accepted.

NONE
- Specifies that no conversion is performed.
The utility fails if RBALRSN_CONVERSION NONE is specified on a table space that is in basic 6-byte format and the UTILITY_OBJECT_CONVERSION subsystem parameter is set to NOBASIC.

**BASIC**

Specifies that if an object is found in extended 10-byte format, it is converted to 6-byte basic format.

The utility fails if RBALRSN_CONVERSION BASIC is specified and the UTILITY_OBJECT_CONVERSION subsystem parameter is set to NOBASIC.

**EXTENDED**

Specifies that if an object is found in basic 6-byte format, it is converted to 10-byte extended format.

If RBALRSN_CONVERSION is not specified, and subsystem parameter UTILITY_OBJECT_CONVERSION is set to BASIC or EXTENDED, RBALRSN_CONVERSION defaults to the UTILITY_OBJECT_CONVERSION value. If RBALRSN_CONVERSION is not specified, and subsystem parameter UTILITY_OBJECT_CONVERSION is set to NOBASIC, RBALRSN_CONVERSION defaults to EXTENDED.

If a CLONE relationship exists, the page set conversion cannot be performed. For clone relationships, you must drop the clone table, convert the base table to extended 10-byte format, and then re-create the clone table.

If AUX YES is also specified, the LOB table spaces and auxiliary indexes are also converted.

Indexes that are rebuilt during REORG TABLESPACE are converted to the same RBA or LRSN format as the indexed table space. REORG TABLESPACE at the PART level converts corresponding partitions of partitioned indexes. Non partitioned indexes are converted if SHRLEVEL CHANGE or REFERENCE is also specified, or if the entire table space is reorganized with SHRLEVEL NONE.

If the 6-byte RBA or LRSN limit has been reached, you might be unable to perform the first insert or load into an XML table space that has XML versioning and that was created with DEFINE NO and basic 6-byte page format. You can run REORG TABLESPACE on the DEFINE NO XML table space to convert its definition to extended 10-byte page format. The REORG must be done on the entire table space.

**DISCARD**

 Specifies that records that meet the specified WHEN conditions are to be discarded during REORG TABLESPACE UNLOAD CONTINUE or UNLOAD PAUSE. If you specify DISCARDDN or a SYSDISC DD statement in the JCL, discarded records are saved in the associated data set. Otherwise, the utility discards records without saving them in a data set.

You can specify any SHRLEVEL option with DISCARD. However, if you specify SHRLEVEL CHANGE, modifications that are made during the reorganization to data rows that match the discard criteria are not permitted. In this case, REORG TABLESPACE terminates with an error.

If you specify DISCARD, rows are decompressed and edit routines are decoded. If you also specify DISCARD to a file, rows are decoded by field procedure, and the following columns are converted to Db2 external format:

- SMALLINT
- INTEGER
- FLOAT
- DECIMAL
- TIME
- TIMESTAMP

Otherwise, edit routines or field procedures are bypassed on both the UNLOAD and RELOAD phases for table spaces. Validation procedures are not invoked during either phase.

**Restrictions:** Do not specify DISCARD if any of the following conditions are true:

- The REORG TABLESPACE statement includes the UNLOAD EXTERNAL or UNLOAD ONLY option.
- The table space to be reorganized is any of the following objects:
  - A base table with XML columns
  - An XML table space
  - A base table with LOB columns if the records to be discarded are more than 32 KB and you want to save them in a data set.
  - A system-period temporal table space

**Related tasks:**
- [Compressing your data (Db2 Performance)](https://www.ibm.com/support/knowledgecenter/SSPS64_11.1.0/com.ibm.db2.luw.admin.sql.ref.doc/id/ck1318.html)

**Related reference:**
- [CANCEL THREAD (Db2) (Db2 Commands)](https://www.ibm.com/support/knowledgecenter/SSPS64_11.1.0/com.ibm.db2.luw.admin.sql.ref.doc/id/ck1330.html)
- Chapter 18, “LISTDEF,” on page 215
- Chapter 34, “TEMPLATE,” on page 793
- [Db2 Sort for z/OS](https://www.ibm.com/support/knowledgecenter/SSS5R6_11.1.0/com.ibm.db2.luw.sql.ref.doc/id/ck1054.html)

**Related information:**

---

**Before running REORG TABLESPACE**

Certain activities might be required before you run the REORG TABLESPACE utility, depending on your situation.

### Catalog and directory table spaces

Before you run REORG on a catalog or directory table space, you must take an image copy. For the DSNDB06.SYSTSCPY catalog table space and the DSNDB01.DBD01 and DSNDB01.SYSDBDXA directory table spaces, REORG scans logs to verify that an image copy is available. If the scan of the logs does not find an image copy, Db2 requests archive logs.

### Region size

The recommended minimum region size is 4096 KB. Region sizes greater than 32 MB enable increased parallelism for index builds. Data unload and reload parallelism can also benefit from a greater region size value.
Mapping tables

If you run REORG TABLESPACE with SHRLEVEL CHANGE, the utility uses a mapping table to store the source and target RID for each row. REORG can implicitly create this mapping table. Alternatively, you can create it yourself.

To specify the database where REORG creates the mapping table, use the MAPPINGDATABASE keyword. If you do not specify a database, REORG uses the value of the REORG_MAPPING_DATABASE subsystem parameter.

If you create the mapping table yourself, follow these guidelines:

- The table space that contains the mapping table must be segmented or partition-by-growth and cannot be the table space to be reorganized.
- To create a segmented table space for the mapping table, use a CREATE TABLESPACE statement similar to the following statement:
  ```sql
  CREATE TABLESPACE table-space-name SEGSIZE integer
  ```
- To create a partition-by-growth table space for the mapping table, use a CREATE TABLESPACE statement similar to the following statement:
  ```sql
  CREATE TABLESPACE table-space-name MAXPARTITIONS integer
  ```
- The mapping table and index can be defined with SQL statements similar to the following statements, substituting other column names as needed:
  ```sql
  CREATE TABLE table-name1
  (TYPE CHAR(1) NOT NULL,
   ORID CHAR(5) NOT NULL,
   NRID CHAR(9) NOT NULL,
   LRSN CHAR(10) NOT NULL)
  IN database-name.table-space-name1;
  CREATE UNIQUE INDEX index-name1 ON table-name1
  (ORID, TYPE, NRID, LRSN);
  ```
- You must specify the NRID column as CHAR(9), even though the RIDs are 5 bytes long.
- These SQL statements can be used in Db2 11 conversion mode or later.
- Beginning in Db2 11 new-function mode, if you create a mapping table with different column specifications and try to specify it in a REORG TABLESPACE statement, the utility does not use it. Instead, REORG TABLESPACE implicitly creates a mapping table in the same database as the mapping table that you specified.
- The size of the table space for the mapping table can be as small as one track. Use the following formula to estimate the minimum number of bytes to allocate for the index on the mapping table:
  ```sql
  1.1 * Number-of-rows-in-table-space * 31
  ```
- You must have DELETE, INSERT, and UPDATE authorization on the mapping table.
- To ensure that multiple REORG jobs do not attempt to use the same mapping table concurrently, consider using the utility ID as the name of the mapping table. Because utility IDs must be unique, this naming convention ensures that the mapping tables are not used by two REORG jobs that run concurrently.

If you run REORG jobs concurrently, each REORG job must have a separate mapping table. The mapping tables do not need to reside in separate table spaces. If more than one REORG job tries to access the same mapping table at the same time, one of the REORG jobs fails. (You can run more than one REORG SHRLEVEL CHANGE job concurrently on separate table spaces. You can also run more than
one REORG SHRLEVEL CHANGE job concurrently on different partitions of the same table space, but only if the table space does not have any nonpartitioned secondary indexes.)

The REORG utility removes all rows from the mapping table when the utility completes.

**Restart-pending status and SHRLEVEL CHANGE**

If you specify SHRLEVEL CHANGE, REORG drains the write claim class near the end of REORG processing. In a data sharing environment, if a data sharing member fails and that member has restart-pending status for a target page set, the drain can fail. You must postpone running REORG with SHRLEVEL CHANGE until all restart-pending statuses are removed. You can use the DISPLAY GROUP command to determine whether a member’s status is failed. You can use the DISPLAY DATABASE command with the LOCKS option to determine if locks are held.

**RECOVER-pending and REBUILD-pending status**

You cannot reorganize a table space if any partition or range of partitions of the partitioned table space is in the RECOVER-pending status. Similarly, you cannot reorganize a single table space partition if any of the following conditions are true:

- The partition is in the RECOVER-pending status.
- The corresponding partitioning index is in the REBUILD-pending or RECOVER-pending status, and the data is unloaded by the cluster index method.
- The specified partition or partitions are a subset of a range of partitions that are in REORG-pending status; you must reorganize the entire range to reset the restrictive status.

The only RECOVER-pending restrictive state is:

**RECP** The table space, index space, or partition of a table space or index space is in a RECOVER-pending status. A single logical partition in RECP does not restrict access to other logical partitions that are not in RECP. You can reset RECP by recovering only the single logical partition.

The three REBUILD-pending restrictive states are:

**RBDP** REBUILD-pending status is set on a physical or logical index partition. The individual physical or logical partition is inaccessible and must be rebuilt by using the REBUILD INDEX utility.

**PSRBD** Page set REBUILD-pending status is set for nonpartitioning indexes. The entire index space is inaccessible and must be rebuilt by using the REBUILD utility.

**RBDP** A REBUILD-pending status that is set only on logical partitions of nonpartitioning indexes. The entire index is inaccessible, but it is made available again when the affected partitions are rebuilt by using the REBUILD INDEX utility.
CHECK-pending status

If a table space is in both REORG-pending and CHECK-pending status (or auxiliary CHECK-pending status), run REORG first, and then run CHECK DATA to clear the respective states. Otherwise, if a table space is not in REORG-pending status, you cannot reorganize a table space or range of partitions if the table space or any partition in the range is in CHECK-pending status until the CHECK-pending status is removed.

REORG-pending status

You must allocate a discard data set (SYSDISC) or specify the DISCARDDN option if the last partition of the table space is in REORG-pending status.

Fallback recovery considerations

If RECOVER cannot use the latest image copy or copies as a starting point for the recovery, it attempts to use previous copies; if that attempt fails, RECOVER restores the data from the log.

However, if you use REORG SHRLEVEL NONE LOG NO, RECOVER cannot restore data from the log past the point at which the object was last reorganized successfully. Therefore, you must take an image copy after running REORG with LOG NO to establish a level of fallback recovery.

Recommendation:

Immediately following an ALTER INDEX operation that modifies key values, create a new recovery point by taking one of the following actions:

• Run REORG and specify COPYDDN and SHRLEVEL NONE.
• Take a full image copy immediately after REORG completes.

If you performed a REORG to reset REORG-pending status (REORP), you should also take an inline image copy or run the COPY utility. Image copies that are taken prior to resetting the REORG-pending status cannot be used for recovery to the current RBA or LRSN.

Successful REORG LOG NO processing inserts a row into SYSIBM.SYSCOPY with ICTYPE=W for each index that was defined with COPY YES. REORG also places a reorganized index in informational COPY-pending (ICOPY) status. You should take a full image copy of the index after the REORG job completes to create a valid point of recovery.

Restrictions when running REORG TABLESPACE on encrypted data

If you plan to run REORG TABLESPACE on column-level encrypted data, do not use the WHEN statement to filter the encrypted fields; REORG TABLESPACE cannot filter rows based on encrypted columns.
Restriction for partitions with the COMPRESS YES attribute when using REBALANCE

Do not run REORG REBALANCE on a partitioned table space where a subset of partitions have the COMPRESS YES attribute and the remaining partitions have the COMPRESS NO attribute.

Restriction when using REBALANCE and duplicate partitioning key values exist

A REORG REBALANCE might distribute rows among the partitions that are being rebalanced in such a way that one or more partitions do not have any rows. This situation occurs when many rows with duplicate partitioning key values exist, and not enough unique values exist to enable REORG to distribute them over all of the partitions.

Restriction for unload parallelism when using REBALANCE

If you specify REORG REBALANCE, you cannot use unload parallelism.

Restrictions for XML table spaces with XML versioning

To REORG an XML table space, with XML versioning, that is in basic 6-byte page format, and that has tables with 8-byte time stamp columns, to extended 10-byte page format, you must REORG the entire table space. It cannot be done at the part level.

Restriction when physical partition numbers do not match logical partition numbers

A REORG REBALANCE might not be possible if the logical and physical partition numbers for the specified table space do not match. This situation can be created by a series of ALTER ROTATEEs and ALTER ADD PARTs.

For example, assume that you create a table space with three partitions. The following table shows the mapping that exists between the physical and logical partition numbers.

Table 80. Mapping of physical and logical partition numbers when a table space with three partitions is created.

<table>
<thead>
<tr>
<th>Logical partition number</th>
<th>Physical partition number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Then, assume that you request the following series of actions:
1. ALTER ROTATE FIRST TO LAST
   
   The new mapping of partition numbers is shown in the following table.

Table 81. Mapping of physical and logical partition numbers after ALTER ROTATE FIRST TO LAST.

<table>
<thead>
<tr>
<th>Logical partition number</th>
<th>Physical partition number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
Table 81. Mapping of physical and logical partition numbers after ALTER ROTATE FIRST TO LAST. (continued)

<table>
<thead>
<tr>
<th>Logical partition number</th>
<th>Physical partition number</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

2. ALTER ADD PART

The new mapping of partition numbers is shown in the following table.

Table 82. Mapping of physical and logical partition numbers after ALTER ADD PART.

<table>
<thead>
<tr>
<th>Logical partition number</th>
<th>Physical partition number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

3. ALTER ROTATE FIRST TO LAST

The new mapping of partition numbers is shown in the following table.

Table 83. Mapping of physical and logical partition numbers after second ALTER ROTATE FIRST TO LAST.

<table>
<thead>
<tr>
<th>Logical partition number</th>
<th>Physical partition number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

Assume that you then try to execute a REORG TABLESPACE REBALANCE PART 1:2. This statement requests a reorganization and rebalancing of physical partitions 1 and 2. Note that physical partition 1 is logical partition 2, and physical partition 2 is logical partition 4. Thus, the utility is processing logical partitions 2 and 4. If during the course of rebalancing, the utility needs to move keys from logical partition 2 to logical partition 3, the job fails, because logical partition 3 is not within the specified physical partition range.

Reorganizing a table space with an index that has a VARBINARY column

If you run REORG against a table space, and that table space includes a table that has an index with the following characteristics, REORG fails:
- The index was created on a VARBINARY column or a column with a distinct type that is based on a VARBINARY data type.
- The index column has the DESC attribute.

To fix the problem, drop the index, or alter the column data type to BINARY, and then rebuild the index.
Potential result of running REORG with many striped inline copy data sets

If a REORG job runs with a large number of striped inline copy data sets, Db2 might not have enough storage for the REORG job to complete successfully. The amount of storage that is needed varies from one job execution to another; therefore, you might have a problem one time and not the next time. Be aware that an ABEND878 can be the result of too many striped inline copy data sets.

Recommendation: If a large number of partitions are being reorganized with striped inline copy data sets and the ABEND878 occurs, divide the REORG job into multiple jobs, each of which handles a subset of the total number of partition ranges. You can also divide the REORG job in this way to avoid the abend altogether.

Related concepts:
- Job DSNTIJ (Db2 Installation and Migration)

Related reference:
- “CHECK-pending status” on page 1039
- “REBUILD-pending status” on page 1043
- “RECOVER-pending status” on page 1044

Data sets that REORG TABLESPACE uses

The REORG TABLESPACE utility uses a number of data sets during its operation.

The following table describes the data sets that REORG TABLESPACE uses. The table lists the DD name that is used to identify the data set, a description of the data set, and an indication of whether it is required. Include statements in your JCL for each required data set, and any optional data sets that you want to use.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNPRINnn</td>
<td>A data set that contains messages from the sort program (usually SYSOUT or DUMMY). This data set is used when distribution statistics are collected for column groups. nn is a number from 01 to the number of parallel subtasks.</td>
<td>No</td>
</tr>
<tr>
<td>SYSIN</td>
<td>Input data set that contains the utility control statement.</td>
<td>Yes</td>
</tr>
<tr>
<td>SYSUT1</td>
<td>A temporary data set for sort input.</td>
<td>No</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>Output data set for messages.</td>
<td>Yes</td>
</tr>
<tr>
<td>STPRIN01</td>
<td>A data set that contains messages from the sort program (usually, SYSOUT or DUMMY). This data set is used when statistics are collected on at least one data-partitioned secondary index, or when COLGROUP and FREQVAL keywords are specified.</td>
<td>Yes</td>
</tr>
<tr>
<td>SYSDISC</td>
<td>Data set that contains discarded records from REORG DISCARD. The default DD name is SYSDISC.</td>
<td>No</td>
</tr>
<tr>
<td>Data set</td>
<td>Description</td>
<td>Required?</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>SYSPUNCH</td>
<td>Data set that contains a LOAD statement that is generated by REORG, which loads records that REORG DISCARD or REORG UNLOAD EXTERNAL wrote to the DISCARD or UNLOAD data sets. The default DD name is SYSPUNCH.</td>
<td>No</td>
</tr>
<tr>
<td>UTPRINT</td>
<td>A data set that contains messages from the sort program (usually, SYSOUT or DUMMY).</td>
<td>Yes</td>
</tr>
<tr>
<td>Unload data set</td>
<td>Data set that contains the unloaded data that is to be reloaded during the RELOAD phase. Specify its DD or template name with the UNLDDN option or with the RECDSN field on the DB2I Utilities panel. The data set must be a sequential data set that is readable by BSAM. The default DD name is SYSREC. The unload data set must be large enough to contain all the unloaded records from all the tables in the target table space. If at least one table in the table space does not have an index, REORG cannot use the SORTDATA method with SHRLEVEL CHANGE. As a result, you must unload the data in the SYSREC data set.</td>
<td>Yes</td>
</tr>
<tr>
<td>Sequential copies</td>
<td>From one to four output data sets that are to contain the image copies. Specify their DD or template names with the COPYDDN and RECOVERYDDN options of the utility control statement.</td>
<td>No</td>
</tr>
<tr>
<td>FlashCopy image copies</td>
<td>For table space or index space level copies, a VSAM data set for the output FlashCopy image copy of each partition or piece. For a partition level or piece level copy, a VSAM data set for the output FlashCopy image copy of the partition or piece.</td>
<td>No</td>
</tr>
<tr>
<td>Work data sets</td>
<td>Temporary data sets for sort input and output. The DD names have the form DATAWKnn.</td>
<td>No</td>
</tr>
<tr>
<td>Work data sets</td>
<td>Temporary data sets for sort input and output when sorting keys, or for sorting data when SORTDATA is specified but NOSYSREC is not. If index build parallelism is used, the DD names have the form SWnnWKmm. If index build parallelism is not used, the DD names have the form SORTWKmm</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Table 84. Data sets that REORG TABLESPACE uses (continued)

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sort work data sets</td>
<td>Temporary data sets for sort input and</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>output when collecting inline statistics on at least one data-partitioned</td>
<td></td>
</tr>
<tr>
<td></td>
<td>secondary index, or when the COLGROUP option or the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>COLGROUP and FREQVAL options are specified. The DD names have the form</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ST01WKnn.</td>
<td></td>
</tr>
<tr>
<td>Sort work data sets</td>
<td>Temporary data sets for sort input and output when collecting inline</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>statistics on at least one data-partitioned secondary index,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>or when the COLGROUP option or the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>COLGROUP and FREQVAL options are specified. The DD names have the form</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DAmmWKmm.</td>
<td></td>
</tr>
<tr>
<td>Sort work data sets</td>
<td>Temporary data sets for sort input and output when collecting distribution</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>statistics for column groups.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The DD names have the form RNmmWKmm, where mm is the subtask number, and</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nn is a sequence number for the data set allocated per task.</td>
<td></td>
</tr>
<tr>
<td>Sort work data sets</td>
<td>Temporary data sets for sort input and output when collecting frequency</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>statistics.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The DD names have the form SORWK01.</td>
<td></td>
</tr>
<tr>
<td>Print data sets</td>
<td>Data sets for unload parallelism.</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>The DD names have the form DTPRINnn.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Every time you invoke REORG TABLESPACE, new DTPRINnn data sets are</td>
<td></td>
</tr>
<tr>
<td></td>
<td>dynamically allocated. REORG TABLESPACE does not reuse DTPRINnn data sets</td>
<td></td>
</tr>
<tr>
<td></td>
<td>from previous job steps. This behavior might cause the available JES2 job</td>
<td></td>
</tr>
<tr>
<td></td>
<td>queue elements to be consumed more quickly than expected.</td>
<td></td>
</tr>
</tbody>
</table>

Note:

1. Required when collecting distribution statistics for column groups
2. STPRIN01 is required if statistics are being collected on at least one data-partitioned secondary index, but REORG TABLESPACE dynamically allocates the STPRIN01 data set if UTPRINT is allocated to SYSOUT.
3. Required when collecting inline statistics on at least one data-partitioned secondary index.
4. Required if you specify DISCARDDDN
5. Required you specify PUNCHDDN
6. Required unless NOSYSREC or SHRLEVEL CHANGE is specified.
7. Required if a partition is in REORG-pending status or REBALANCE, COPYDDN, RECOVERYDDN, SHRLEVEL REFERENCE, or SHRLEVEL CHANGE is specified.
8. Required if NOSYSREC or SHRLEVEL CHANGE is specified, but SORTDEVT is not specified.
9. Required if any indexes exist and SORTDEVT is not specified.
10. If the DYNALLOC parm of the SORT program is not turned on, you need to allocate the data set. Otherwise, the sort program dynamically allocates the temporary data set.
11. If you specify the SORTDEVT keyword, the data sets are dynamically allocated. It is recommended that you use dynamic allocation by specifying SORTDEVT in the utility statement because dynamic allocation reduces the maintenance required of the utility job JCL.
12. If UTPRINT is allocated to SYSOUT, the data sets are dynamically allocated.
13. Required if you specify either FLASHCOPY YES or FLASHCOPY CONSISTENT.
14. Required when the COLGROUP and FREQVAL options are specified.
The following objects are named in the utility control statement and do not require DD statements in the JCL:

**Table space**
Object that is to be reorganized.

### Calculating the size of the unload data set

The required size for the unload data set varies depending on the options that you use for REORG.

1. **If you use REORG with UNLOAD PAUSE or CONTINUE and you specify KEEPDICTIONARY** (assuming that a compression dictionary already exists), the size of the unload data set, in bytes, is the VSAM high-allocated RBA for the table space. You can obtain the high-allocated RBA from the associated VSAM catalog.

   For SHRLEVEL CHANGE, also add the result of the following calculation (in bytes) to the VSAM high-used RBA:
   
   \[
   \text{number of records} \times 11
   \]

2. **If you use REORG with UNLOAD ONLY, UNLOAD PAUSE, or CONTINUE and you do not specify KEEPDICTIONARY, you can calculate the size of the unload data set, in bytes, by using the following formula:**

   \[
   \text{maximum row length} \times \text{number of rows}
   \]

   The maximum row length is the row length, including the 6-byte record prefix, plus the length of the longest clustering key. If multiple tables exist in the table space, use the following formula to determine the maximum row length:

   \[
   \text{Sum over all tables } \left( \text{row length} + (2 \times \text{number of VARBIN columns}) \times \text{number of rows} \right)
   \]

   For SHRLEVEL CHANGE, also add the result of the following formula to the preceding result:

   \[
   (21 \times \left( \text{NEARINDREF} + \text{FARINDREF} \right) \times 1.1)
   \]

   In the preceding formula:

   **NEARINDREF**
   Is the value that is obtained from the NEARINDREF column of the SYSIBM.SYSTABLEPART catalog table. The accuracy of the data set size calculation depends on recent information in the SYSTABLEPART catalog table.

   **FARINDREF**
   Is the value that is obtained from the FARINDREF column of the SYSIBM.SYSTABLEPART catalog table.

3. **If you have variable-length fields, the calculation in step 2 might result in excessive space. Use the average uncompressed row length, multiplied by the number of rows.**

4. **If you use REORG with UNLOAD PAUSE or CONTINUE with the DISCARD option, and the table has variable length fields, use the maximum row length in the calculation. The DISCARD option without the NOPAD option pads the variable length fields.**
For certain table spaces in the catalog and directory, the unload data set for the table spaces have a different format. The calculation for the size of this data set is as follows:

\[
\text{data set size in bytes} = (28 + \text{longrow}) \times \text{numrows}
\]

In the preceding formula:

- \text{longrow}
  - Is the length of the longest row in the table space.

- \text{numrows}
  - Is the number of rows in the data set.

The length of the row is calculated as follows:

\[
\text{Sum of column lengths} + 4 \text{ bytes for each link}
\]

The length of the column is calculated as follows:

\[
\text{Maximum length of the column} + 1 \text{ (if nullable)} + 2 \text{ (if varying length)}
\]

**Calculating the size of the sort work data sets**

Using two or three large SORTWKnn data sets is preferable to using several small ones. If adequate space is not available, you cannot run REORG.

Sort work data sets cannot span volumes. Smaller volumes require more sort work data sets to sort the same amount of data; therefore, large volume sizes can reduce the number of sort work data sets that are needed.

When you allocate sort work data sets on disk, allocate at least 1.2 times the amount of space that is used by the data that is to be sorted.

Allocating twice the space that is used by the unload data sets is usually adequate for the sort work data sets. For compressed data, if the data needs to be uncompressed for processing, allocate additional sort space. Use the compression ratio of the existing data to calculate the additional sort space that is required. Some examples of when the compressed data needs to be uncompressed include, but are not limited to:

- UNLOAD PAUSE without KEEPDICTIONARY
- UNLOAD CONTINUE without KEEPDICTIONARY
- Discard processing
- After schema changes, such as ALTER ADD COLUMN
- When AUX YES processing is in effect

Tape devices are not supported for sort work data sets.

**Specifying a destination for sort program messages**

The REORG utility job step must contain a UTPRINT DD statement that defines a destination for messages that are issued by the sort program during the SORT phase of REORG. DB2I, the %DSNU CLIST command, and the DSNUPROC procedure use the following default DD statement:

```
//UTPRINT DD SYSOUT=A
```
Calculating the size of the statistics sort work data sets:

To calculate the approximate size (in bytes) of the ST01WKnn data set, use the following formula:

\[ 2 \times (\text{maximum record length} \times \text{numcols} \times (\text{count} + 2) \times \text{number of indexes}) \]

The variables in the preceding formula have the following values:

- **maximum record length**: Maximum record length of the SYSCOLDISTSTATS record that is processed when collecting frequency statistics (You can obtain this value from the RECLENGTH column in SYSTABLES.)
- **numcols**: Number of key columns to concatenate when you collect frequent values from the specified index.
- **count**: Number of frequent values that Db2 is to collect.

Related concepts:

- “Reorganizing the catalog and directory” on page 628
- “Data sets that online utilities use” on page 16

Related reference:

- [Db2 Sort for z/OS](utility_guide_and_reference)

Related information:

- [DFSORT Application Programming Guide](utility_guide_and_reference)

Shadow data sets for REORG TABLESPACE

When you execute the REORG utility with SHRLEVEL REFERENCE or SHRLEVEL CHANGE, the utility uses shadow data sets.

For user-managed data sets, you must preallocate the shadow data sets before you execute REORG with SHRLEVEL REFERENCE or SHRLEVEL CHANGE. If a table space, partition, or index resides in Db2-managed data sets and shadow data sets do not already exist when you execute REORG, Db2 creates the shadow data sets. At the end of REORG processing, the Db2-managed shadow data sets are deleted.

Shadow data set names

Each shadow data set must have the following name:

\[ \text{catname.DSNDBx.dbname.psname.y000z.Lnnn} \]

In the preceding name, the variables have the following meanings:

- **variable**
  - **meaning**
  - **catname**: The VSAM catalog name or alias
  - \( x \): C or D
  - **dbname**: Database name
  - **psname**: Table space name or index name
I or J

z 1 or 2

$Lnnn$ Partition identifier. Use one of the following values:

- A001 through A999 for partitions 1 through 999
- B000 through B999 for partitions 1000 through 1999
- C000 through C999 for partitions 2000 through 2999
- D000 through D999 for partitions 3000 through 3999
- E000 through E996 for partitions 4000 through 4096

To determine the names of existing data sets, execute one of the following queries against the SYSTABLEPART or SYSINDEXPART catalog tables:

```
SELECT DBNAME, TSNAME, IPREFIX FROM SYSIBM.SYSTABLEPART
  WHERE DBNAME = 'dbname'
    AND TSNAME = 'psname';

SELECT IXNAME, PARTITION, IPREFIX FROM SYSIBM.SYSINDEXPART
  WHERE IXNAME = 'ixname'
    AND IXCREATOR = 'creator'
    AND IXDBNAME = 'dbname'
    AND IXINDEXSPACE = 'psname';
```

For a partitioned table space, Db2 returns rows from which you select the row for the partitions that you want to reorganize.

For example, assume that you have a ten-partition table space and you want to determine a naming convention for the data set in order to successfully execute the REORG utility with the SHRLEVEL CHANGE PART 2:6 options. The following queries of the Db2 catalog tables SYSTABLEPART and SYSINDEXPART provide the required information:

```
SELECT DBNAME, TSNAME, PARTITION, IPREFIX FROM SYSIBM.SYSTABLEPART
  WHERE DBNAME = 'db0v0701' AND TSNAME = 'tp0v0701'
    ORDER BY PARTITION;

SELECT IXNAME, PARTITION, IPREFIX FROM SYSIBM.SYSINDEXPART
  WHERE IXNAME = 'idxv0701'
    ORDER BY PARTITION;
```

The preceding queries produce the information that is shown in the following table.

The following table shows the results from the first query.
Table 85. Query results from the first preceding query

<table>
<thead>
<tr>
<th>DBNAME</th>
<th>TSNAME</th>
<th>PARTITION</th>
<th>IPREFIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBDV0701</td>
<td>TPDV0701</td>
<td>1</td>
<td>I</td>
</tr>
<tr>
<td>DBDV0701</td>
<td>TPDV0701</td>
<td>4</td>
<td>I</td>
</tr>
<tr>
<td>DBDV0701</td>
<td>TPDV0701</td>
<td>3</td>
<td>J</td>
</tr>
<tr>
<td>DBDV0701</td>
<td>TPDV0701</td>
<td>2</td>
<td>I</td>
</tr>
<tr>
<td>DBDV0701</td>
<td>TPDV0701</td>
<td>5</td>
<td>J</td>
</tr>
<tr>
<td>DBDV0701</td>
<td>TPDV0701</td>
<td>6</td>
<td>J</td>
</tr>
<tr>
<td>DBDV0701</td>
<td>TPDV0701</td>
<td>7</td>
<td>I</td>
</tr>
<tr>
<td>DBDV0701</td>
<td>TPDV0701</td>
<td>8</td>
<td>I</td>
</tr>
<tr>
<td>DBDV0701</td>
<td>TPDV0701</td>
<td>9</td>
<td>I</td>
</tr>
<tr>
<td>DBDV0701</td>
<td>TPDV0701</td>
<td>10</td>
<td>I</td>
</tr>
</tbody>
</table>

The following table shows the results from the second query.

Table 86. Query results from the second preceding query

<table>
<thead>
<tr>
<th>IXNAME</th>
<th>PARTITION</th>
<th>IPREFIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>IXDV0701</td>
<td>10</td>
<td>I</td>
</tr>
<tr>
<td>IXDV0701</td>
<td>9</td>
<td>I</td>
</tr>
<tr>
<td>IXDV0701</td>
<td>8</td>
<td>I</td>
</tr>
<tr>
<td>IXDV0701</td>
<td>7</td>
<td>I</td>
</tr>
<tr>
<td>IXDV0701</td>
<td>6</td>
<td>J</td>
</tr>
<tr>
<td>IXDV0701</td>
<td>5</td>
<td>J</td>
</tr>
<tr>
<td>IXDV0701</td>
<td>4</td>
<td>I</td>
</tr>
<tr>
<td>IXDV0701</td>
<td>3</td>
<td>J</td>
</tr>
<tr>
<td>IXDV0701</td>
<td>2</td>
<td>I</td>
</tr>
<tr>
<td>IXDV0701</td>
<td>1</td>
<td>I</td>
</tr>
</tbody>
</table>

To execute REORG SHRLEVEL CHANGE PART 2:6, you need to preallocate the following shadow objects. The naming convention for these objects use information from the query results that are shown in the previous tables.

- vcatnam.DSNDBC.DBDV0701.TPDV0701.J0001.A002
- vcatnam.DSNDBC.DBDV0701.TPDV0701.10001.A003
- vcatnam.DSNDBC.DBDV0701.TPDV0701.10001.A004
- vcatnam.DSNDBC.DBDV0701.TPDV0701.10001.A005
- vcatnam.DSNDBC.DBDV0701.TPDV0701.10001.A006
- vcatnam.DSNDBC.DBDV0701.IXDV0701.J0001.A002
- vcatnam.DSNDBC.DBDV0701.IXDV0701.10001.A003
- vcatnam.DSNDBC.DBDV0701.IXDV0701.10001.A004
- vcatnam.DSNDBC.DBDV0701.IXDV0701.10001.A005
- vcatnam.DSNDBC.DBDV0701.IXDV0701.10001.A006

Defining shadow data sets

Consider the following actions when you preallocate the data sets:

- Allocate the shadow data sets according to the rules for user-managed data sets.
- Define the shadow data sets as LINEAR.
- Use SHAREOPTIONS(3,3).
• Define the shadow data sets as EA-enabled if the original table space or index space is EA-enabled.
• Allocate the shadow data sets on the volumes that are defined in the storage group for the original table space or index space.

If you specify a secondary space quantity, Db2 does not use it. Instead, Db2 uses the SECQTY value for the table space or index space.

**Recommendation:** Use the MODEL option, which causes the new shadow data set to be created like the original data set. This method is shown in the following example:

```
DEFINE CLUSTER +
  (NAME('catname.DSNDBC.dbname.psname.x0001.L001') +
   MODEL('catname.DSNDBC.dbname.psname.y0001.L001')) +
DATA +
  (NAME('catname.DSNDBD.dbname.psname.x0001.L001') +
   MODEL('catname.DSNDBD.dbname.psname.y0001.L001'))
```

**Creating shadow data sets for indexes:**

When you preallocate data sets for indexes, create the shadow data sets as follows:

• Create shadow data sets for the partition of the table space and the corresponding partition in each partitioning index and data-partitioned secondary index.
• Create a shadow data set for each nonpartitioned secondary index.

Use the same naming scheme for these index data sets as you use for other data sets that are associated with the base index, except use J0001 instead of I0001. For more information about this naming scheme, see the information about the shadow data set naming convention at the beginning of this topic.

**Estimating the size of shadow data sets**

If you have not changed the value of FREEPAGE or PCTFREE, the amount of required space for a shadow data set is comparable to the amount of required space for the original data set.

**Preallocating shadow data sets for REORG PART**

By creating the shadow data sets before executing REORG PART, even for Db2-managed data sets, you prevent possible over-allocation of the disk space during REORG processing. When reorganizing a partition, you must create the shadow data sets for the partition of the table space and for the partition of the partitioning index. In addition, before executing REORG PART with SHRLEVEL REFERENCE or SHRLEVEL CHANGE on partition `mmm` of a partitioned table space, you must create a shadow data set for each nonpartitioning index that resides in user-defined data sets. Each shadow data set is to be used for a copy of the index and must be as large as the entire original nonpartitioned index. The name for this shadow data set has the form `catname.DSNDBx.dbname.psname.y0mmm.Annn`.

**Concurrency and compatibility for REORG TABLESPACE**

The REORG TABLESPACE utility has certain concurrency and compatibility characteristics associated with it.
Db2 treats individual data and index partitions, and individual logical partitions of nonpartitioning indexes as distinct target objects. Utilities that operate on different partitions of the same table space or index space are compatible. However, REORG SHRLEVEL CHANGE or REFERENCE on a partition or range of partitions rebuild entire nonpartitioned indexes; therefore, two REORG SHRLEVEL CHANGE or REFERENCE PART jobs on different partitions of the same table space are not compatible.

**Restriction:** You cannot run concurrent REORG TABLESPACE SHRLEVEL CHANGE PART integer on the same table space. Instead of submitting multiple jobs, you can merge the jobs into one job and specify a range using REORG TABLESPACE SHRLEVEL CHANGE PART integer1:integer2, or specify REORG TABLESPACE SHRLEVEL CHANGE SCOPE PENDING if multiple partitions are in a REORG-pending state.

This information includes a series of tables that show which claim classes REORG drains and any restrictive state that the utility sets on the target object.

For nonpartitioned indexes, if you specify SHRLEVEL NONE, REORG PART:
- Drains only the logical partition (and the repeatable read class for the entire index)
- Does not set the page set REBUILD-pending status (PSRCP)
- Does not use PCTFREE or FREETABLE attributes when inserting keys

### Claim classes that REORG TABLESPACE drains

For SHRLEVEL NONE, the following table shows which claim classes REORG drains and any restrictive state that the utility sets on the target object. For each column, the table indicates the claim or drain that is acquired and the restrictive state that is set in the corresponding phase. UNLOAD CONTINUE and UNLOAD PAUSE, unlike UNLOAD ONLY, include the RELOAD phase and thus include the drains and restrictive states of that phase.

<table>
<thead>
<tr>
<th>Target</th>
<th>UNLOAD phase of REORG</th>
<th>RELOAD phase of REORG if UNLOAD CONTINUE or PAUSE</th>
<th>UNLOAD phase of REORG PART</th>
<th>RELOAD phase of REORG PART if UNLOAD CONTINUE or PAUSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table space, partition, or a range of partitions of a table space</td>
<td>DW/UTRO</td>
<td>DA/UTUT</td>
<td>DW/UTRO</td>
<td>DA/UTUT</td>
</tr>
<tr>
<td>Partitioning index, data-partitioned secondary index, or partition of either type of index¹</td>
<td>DW/UTRO</td>
<td>DA/UTUT</td>
<td>DW/UTRO</td>
<td>DA/UTUT</td>
</tr>
<tr>
<td>Nonpartitioned index²</td>
<td>DW/UTRO</td>
<td>DA/UTUT</td>
<td>None</td>
<td>DR</td>
</tr>
<tr>
<td>Logical partition of nonpartitioning index³</td>
<td>None</td>
<td>None</td>
<td>DW/UTRO</td>
<td>DA/UTUT</td>
</tr>
</tbody>
</table>
Table 87. Claim classes of REORG TABLESPACE SHRLEVEL NONE operations (continued)

<table>
<thead>
<tr>
<th>Target</th>
<th>UNLOAD phase of REORG</th>
<th>RELOAD phase of REORG if UNLOAD CONTINUE or PAUSE</th>
<th>UNLOAD phase of REORG PART</th>
<th>RELOAD phase of REORG PART if UNLOAD CONTINUE or PAUSE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:
- DA: Drain all claim classes, no concurrent SQL access.
- DR: Drain the repeatable read class, no concurrent access for SQL repeatable readers.
- DW: Drain the write claim class, concurrent access for SQL readers.
- UTUT: Utility restrictive state, exclusive control.
- UTRO: Utility restrictive state, read-only access allowed.
- None: Any claim, drain, or restrictive state for this object does not change in this phase.

Note:
1. Includes document ID indexes and node ID indexes over partitioned XML table spaces.
2. Includes document ID indexes and node ID indexes over nonpartitioned XML table spaces and XML indexes.
3. Includes logical partitions of an XML index over partitioned XML table spaces.

For SHRLEVEL REFERENCE, the following table shows which claim classes REORG drains and any restrictive state that the utility sets on the target object. For each column, the table indicates the claim or drain that is acquired and the restrictive state that is set in the corresponding phase.

Table 88. Claim classes of REORG TABLESPACE SHRLEVEL REFERENCE operations

<table>
<thead>
<tr>
<th>Target</th>
<th>UNLOAD phase of REORG</th>
<th>SWITCH phase of REORG</th>
<th>UNLOAD phase of REORG PART</th>
<th>SWITCH phase of REORG PART</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table space or partition of table space</td>
<td>DW/UTRO</td>
<td>DA/UTUT</td>
<td>DW/UTRO</td>
<td>DA/UTUT</td>
</tr>
<tr>
<td>Partitioning index, data-partitioned secondary index, or partition of either1</td>
<td>DW/UTRO</td>
<td>DA/UTUT</td>
<td>DW/UTRO</td>
<td>DA/UTUT</td>
</tr>
<tr>
<td>Nonpartitioned secondary index2</td>
<td>DW/UTRO</td>
<td>DA/UTUT</td>
<td>CR/UTRW</td>
<td>DA/UTUT</td>
</tr>
<tr>
<td>Logical partition of nonpartitioning index3</td>
<td>None</td>
<td>None</td>
<td>DW/UTRO</td>
<td>DA/UTUT</td>
</tr>
</tbody>
</table>

Legend:
- CR: Claim read, concurrent access for SQL writers and readers
- DA: Drain all claim classes, no concurrent SQL access.
- DW: Drain the write claim class, concurrent access for SQL readers.
- UTUT: Utility restrictive state, exclusive control.
- UTRO: Utility restrictive state, read-only access allowed.
- UTRW: Utility restrictive state, read-write access allowed.
- None: Any claim, drain, or restrictive state for this object does not change in this phase.

Note:
1. Includes document ID indexes and node ID indexes over partitioned XML table spaces.
2. Includes document ID indexes and node ID indexes over nonpartitioned XML table spaces and XML indexes.
3. Includes logical partitions of an XML index over partitioned XML table spaces.
For REORG of an entire table space with SHRLEVEL CHANGE, the following table shows which claim classes REORG drains and any restrictive state that the utility sets on the target object.

Table 89. Claim classes of REORG TABLESPACE SHRLEVEL CHANGE operations

<table>
<thead>
<tr>
<th>Target</th>
<th>UNLOAD phase</th>
<th>Last iteration of LOG phase</th>
<th>SWITCH phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table space</td>
<td>CR/UTRW⁴</td>
<td>DW/UTRO</td>
<td>DA/UTUT</td>
</tr>
<tr>
<td>Index</td>
<td>CR/UTRW⁴</td>
<td>DW/UTRO</td>
<td>DA/UTUT</td>
</tr>
</tbody>
</table>

Legend:
- CR: Claim the read claim class.
- DA: Drain all claim classes, no concurrent SQL access.
- DW: Drain the write claim class, concurrent access for SQL readers.
- UTUT: Utility restrictive state, exclusive control.
- UTRO: Utility restrictive state, read-only access allowed.
- UTRW: Utility restrictive state, read-write access allowed.

For REORG of a partition with SHRLEVEL CHANGE, the following table shows which claim classes REORG drains and any restrictive state that the utility sets on the target object.

Table 90. Claim classes of REORG TABLESPACE SHRLEVEL CHANGE operations on a partition

<table>
<thead>
<tr>
<th>Target</th>
<th>UNLOAD phase</th>
<th>Last iteration of LOG phase</th>
<th>SWITCH phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition of table space</td>
<td>CR/UTRW</td>
<td>DW/UTRO or DA/UTUT¹</td>
<td>DA/UTUT</td>
</tr>
<tr>
<td>Partition of partitioned index</td>
<td>CR/UTRW</td>
<td>DW/UTRO or DA/UTUT¹</td>
<td>DA/UTUT</td>
</tr>
<tr>
<td>Non-partitioned index</td>
<td>CR/UTRW</td>
<td>DW/UTRO or DA/UTUT¹</td>
<td>DA/UTUT</td>
</tr>
</tbody>
</table>

Legend:
- CR: Claim the read claim class.
- DA: Drain all claim classes, no concurrent SQL access.
- DDR: Dedrain the read claim class, no concurrent access for SQL repeatable readers.
- DR: Drain the repeatable read class, no concurrent access for SQL repeatable readers.
- DW: Drain the write claim class, concurrent access for SQL readers.
- UTUT: Utility restrictive state, exclusive control.
- UTRO: Utility restrictive state, read-only access allowed.
- UTRW: Utility restrictive state, read-write access allowed.
- None: Any claim, drain, or restrictive state for this object does not change in this phase.

Note:
1. DA/UTUT applies if you specify DRAIN ALL.

Compatibility of REORG TABLESPACE with other utilities

The following table shows which utilities can run concurrently with REORG on the same target object. The target object can be a table space, an index space, or a partition of a table space or index space. If compatibility depends on particular options of a utility, that information is also shown.
<table>
<thead>
<tr>
<th>Action</th>
<th>REORG SHRLEVEL NONE UNLOAD CONTINUE or PAUSE, REORG SHRLEVEL REFERENCE, or REORG SHRLEVEL CHANGE</th>
<th>REORG SHRLEVEL NONE UNLOAD ONLY without clustering index</th>
<th>REORG SHRLEVEL NONE UNLOAD ONLY with clustering index</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATMAINT</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CHECK DATA</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CHECK INDEX</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CHECK LOB</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>COPY INDEXSPACE</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COPY TABLESPACE</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COPYTOCOPY</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DIAGNOSE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LOAD</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MERGECOPY</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MODIFY RECOVERY</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MODIFY STATISTICS</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>QUIESCE</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>REBUILD INDEX</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>RECOVER INDEX</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>RECOVER INDEXSPACE</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RECOVER TABLESPACE</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REORG INDEX</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>REORG TABLESPACE SHRLEVEL NONE UNLOAD CONTINUE or PAUSE, REORG SHRLEVEL REFERENCE, or REORG SHRLEVEL CHANGE</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REORG TABLESPACE SHRLEVEL NONE UNLOAD ONLY or EXTERNAL</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>REPAIR DUMP or VERIFY</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>REPAIR LOCATE KEY or RID DELETE or REPLACE</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REPAIR LOCATE INDEX PAGE REPLACE</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>REPAIR LOCATE TABLESPACE PAGE REPLACE</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REPORT</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RUNSTATS</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>STOSPACE</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UNLOAD</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
The following table shows which Db2 operations can be affected when reorganizing catalog table spaces.

**Table 92. Db2 operations that are affected by reorganizing catalog table spaces**

<table>
<thead>
<tr>
<th>Catalog table space</th>
<th>Actions that might not run concurrently</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any table space except SYSTSCPY, SYSTSCHX, SYSTSCKD, SYSTSSRG, and SYSTSCKS</td>
<td>CREATE, ALTER, and DROP statements</td>
</tr>
<tr>
<td>SYSTSCPY, SYSTSFAU, SYSTSCOL, SYSTSTSP, SYSTSTPT, SYSTSTAB, SYSTSIXS, SYSTSIXT, SYSTSIXR, SYSTSIPT, SYSTSRGL, SYSTSFOR, SYSTSSYN, SYSTSFLD, SYSTSTAU, SYSTSDBA, SYSTSDBU, SYSTSK1Y, SYSTSDBA, SYSTSTSG, SYSTSTTS, SYSTSSUSER, SYSTSIV</td>
<td>Utilities</td>
</tr>
<tr>
<td>SYSTSFU, SYSTSCOL, SYSTSTSP, SYSTSTPT, SYSTSTAB, SYSTSIXS, SYSTSIXT</td>
<td>GRANT and REVOKE statements</td>
</tr>
<tr>
<td>SYSTSFU, SYSTSCOL, SYSTSTSP, SYSTSTPT, SYSTSTAB, SYSTSIXS, SYSTSIXT, SYSTSIXR, SYSTSIPT, SYSTSRGL, SYSTSFOR, SYSTSSYN, SYSTSFLD, SYSTSTAU, SYSTSDBA, SYSTSDBU, SYSTSK1Y, SYSTSDBA, SYSTSTSG, SYSTSTTS, SYSTSSUSER, SYSTSIV</td>
<td>BIND and FREE commands</td>
</tr>
</tbody>
</table>

**Notes:**
1. Most utilities that register their execution in SYSTSCPY can run concurrently with REORG SHRLEVEL CHANGE of SYSTSCPY.

**Determining whether an object requires reorganization**

You must reorganize an object if it is in the REORG-pending (REORP) restrictive status. Also, consider reorganizing an object if it is in an advisory REORG-pending status (AREO* or AREOR) or if analysis shows that reorganization might improve performance. Use the REORG INDEX or REORG TABLESPACE utility to reorganize the object.

**About this task**

**Recommendation:** Run the RUNSTATS utility if the statistics are not current. If the object should also be reorganized, run REORG with STATISTICS and take inline
copies. If you run REORG PART and nonpartitioning indexes exist, subsequently run RUNSTATS for each nonpartitioning index.

**Procedure**

To determine whether an object requires reorganization, use any of the following approaches:

- Reorganize table spaces or partitions that are in REORG-pending status. Use the `DISPLAY DATABASE RESTRICT` command to display those table spaces and partitions that require reorganization.
- Run the REORG TABLESPACE utility and specify the `OFFPOSLIMIT` and `INDREFLIMIT` catalog query options with the `REPORTONLY` option. REORG produces a report with one of the following return codes, but the object is not reorganized.
  1. No limit met; no reorganization is performed or recommended.
  2. A reorganization is performed or recommended.
- Use the SYSTABLEPART and SYSINDEXPART catalog tables to find which table spaces and indexes qualify for reorganization. The information in these catalog tables can also be used to determine when the Db2 catalog table spaces require reorganization.

Information from the SYSTABLEPART catalog table can also indicate how well disk space is being used. If you want to find the number of varying-length rows that were relocated to other pages because of an update, run RUNSTATS, and then issue the following statement:

```sql
SELECT CARD, NEARINDREF, FARINDREF
FROM SYSIBM.SYSTABLEPART
WHERE DBNAME = 'XXX'
  AND TSNAME = 'YYY';
```

A large number (relative to previous values that you received) for `FARINDREF` indicates that I/O activity on the table space is high. If you find that this number increases over a time, you probably need to reorganize the table space to improve performance. You probably also need to increase `PCTFREE` or `FREEPAGE` for the table space with the `ALTER TABLESPACE` statement.

The following statement returns the percentage of unused space in nonsegmented table space `YYY`. In nonsegmented table spaces, the space that is used by dropped tables is not reclaimed until you reorganize the table space.

```sql
SELECT PERCDROP
FROM SYSIBM.SYSTABLEPART
WHERE DBNAME = 'XXX'
  AND TSNAME = 'YYY';
```

Issue the following statement to determine whether the rows of a table are stored in the same order as the entries of its clustering index:
SELECT NEAROFFPOSF, FAROFFPOSF
FROM SYSIBM.SYSINDEXPART
    WHERE IXCREATOR = 'index_creator_name'
    AND IXNAME = 'index_name';

Several indicators are available to signal a time for reorganizing table spaces. A large value for FAROFFPOSF might indicate that clustering is deteriorating. In this case, reorganize the table space to improve query performance.

A large value for NEAROFFPOSF might indicate also that reorganization might improve performance. However, in general NEAROFFPOSF is not as critical a factor as FAROFFPOSF.

**What to do next**

For any table, the REORG utility repositions rows into the sequence of the key of the clustering index that is defined on that table.

For nonclustering indexes, the statistical information that is recorded by RUNSTATS in SYSINDEXES and SYSINDEXPART might be even worse after the clustering index is used to reorganize the data. This situation applies only to the CLUSTERING and CLUSTERED columns in SYSINDEXES and to the NEAROFFPOS and FAROFFPOS columns in SYSINDEXPART.

**Related tasks:**

- Maintaining data organization and statistics (Db2 Performance)

**Related reference:**

- Chapter 32, “RUNSTATS,” on page 741
- SYSCBM.SYSTABLEPART table (Db2 SQL)
- SYSCBM.SYSINDEXES table (Db2 SQL)
- SYSCBM.SYSINDEXPART table (Db2 SQL)
- “REORG-pending status” on page 1045

**Access with REORG TABLESPACE SHRLEVEL**

You can specify the level of access that you have to your data by using the SHRLEVEL option.

For reorganizing a table space, or a partition of a table space, the SHRLEVEL option lets you choose the level of access that you have to your data during reorganization.

- REORG with SHRLEVEL NONE, the default, reloads the reorganized data into the original area that is being reorganized. Applications have read-only access during unloading and no access during reloading. For data-partitioned secondary indexes, the option rebuilds the index parts during the BUILD phase. (Rebuilding these indexes does not create contention between parallel REORG PART jobs.) For nonpartitioned secondary indexes, the option corrects the indexes.
• REORG with SHRLEVEL REFERENCE reloads the reorganized data into a new (shadow) copy of the area that is being reorganized. Near the end of reorganization, Db2 switches the future access of the application from the original data to the shadow copy. For SHRLEVEL REFERENCE, applications have read-only access during unloading and reloading, and a brief period of no access during switching.

• REORG with SHRLEVEL CHANGE reloads the reorganized data into a shadow copy of the area that is being reorganized. For REORG TABLESPACE SHRLEVEL CHANGE, a mapping table correlates RIDs in the original copy of the table space or partition with RIDs in the shadow copy. Applications can read from and write to the original area, and Db2 records the writing in the log. Db2 then reads the log and applies it to the shadow copy to bring the shadow copy up to date. This step executes iteratively, with each iteration processing a sequence of log records. Near the end of reorganization, Db2 switches the future access of the application from the original data to the shadow copy. Applications have read-write access during unloading and reloading, a brief period of read-only access during the last iteration of log processing, and a brief period of no access during switching.

• REORG TABLESPACE SHRLEVEL CHANGE and COPY SHRLEVEL CHANGE are compatible and can run concurrently except during the period when exclusive control is needed to drain claimers of a target table space.

Restriction:

– COPY with the FLASHCOPY CONSISTENT option is not compatible with REORG.

– If REORG has drained the claimers of a table space or table space partition and a COPY utility is submitted to access the same object, the COPY utility terminates with a message that it is not compatible.

– If COPY and REORG are accessing the same table space or table space partitions, REORG cannot drain claimers until COPY completes. The REORG DRAIN options determine the actions taken.

– If COPY and REORG are accessing the same table space or table space partitions and COPY abends, restart of the COPY is not allowed if REORG completes.

• REORG TABLESPACE with SHRLEVEL REFERENCE or SHRLEVEL CHANGE materializes pending definition changes for table spaces and indexes if pending alterations are involved. Advisory-REORG pending status (AREOR) is reset for the table space and associated indexes. REORG TABLESPACE with SHRLEVEL REFERENCE or SHRLEVEL CHANGE at the partition level does not materialize pending definition changes at the table space level.

REORG TABLESPACE with SHRLEVEL NONE proceeds without materializing pending definition changes if there were any on the object being reorganized. When pending definition changes are materialized during REORG TABLESPACE with SHRLEVEL REFERENCE or SHRLEVEL CHANGE, statistics for both table space and associated indexes are collected and updated in the Db2 catalog.

• REORG TABLESPACE with SHRLEVEL REFERENCE or SHRLEVEL CHANGE drops empty partitions from a partition-by-growth table space if the REORG_DROP_PBG_PPARTS subsystem parameter is set to ENABLE. With ENABLE, during the UTILTERM phase, REORG cancels all claimers in order to drop the empty partitions, regardless of the value specified for the FORCE keyword. If the REORG_DROP_PBG_PPARTS subsystem parameter is set to DISABLE, REORG TABLESPACE does not drop empty partitions.
• **REORG TABLESPACE with the SHRLEVEL REFERENCE and REBALANCE**
options does not materialize pending definition changes for conversion of a
partitioned table space to partition-by-range table space.

• **REORG TABLESPACE with SHRLEVEL REFERENCE or SHRLEVEL CHANGE**
and the FASTSWITCH NO option does not materialize pending definition
changes.

• When **REORG TABLESPACE with SHRLEVEL REFERENCE or SHRLEVEL CHANGE** is run with the AUX YES option on an entire base table space of one
of the following types, the pending changes that are associated with the base
table space are materialized, but the pending changes that are associated with
the LOB table spaces are not materialized.
  – Simple table space
  – Segmented (non-UTS) table space
  – partition-by-range table space
  – partition-by-growth table space

• When **REORG TABLESPACE with SHRLEVEL REFERENCE or SHRLEVEL CHANGE** is run with the AUX YES option on a subset of partitions of a
partitioned table base table space, neither the pending changes that are associated with the base table space nor the pending changes that are associated with the LOB table spaces are materialized.

• If large amounts of data are deleted from a partition-by-growth table space,
including XML table spaces, run the **REORG TABLESPACE** utility with
SHRLEVEL REFERENCE or SHRLEVEL CHANGE on the entire table space to
reclaim physical space from the partition-by-growth table space.

• After **RECOVER** is run to recover a table space to a point in time before the
materialization of pending definition changes, the entire table space or affected
partitions are placed in REORG-pending (REORP) status. **REORG TABLESPACE**
with SHRLEVEL REFERENCE must be run on the entire table space or affected
partitions to remove REORG-pending status and to complete the point-in-time
recovery process.

**Related information:**


**Log processing with SHRLEVEL REFERENCE PART for nonpartitioned indexes or SHRLEVEL CHANGE:**

When you specify SHRLEVEL REFERENCE PART for nonpartitioned indexes or
SHRLEVEL CHANGE, Db2 processes the log to update the shadow copy. This step
executes iteratively. The first iteration processes the log records that accumulated
during the previous iteration. The iterations continue until one of these conditions
is met:

• Db2 estimates that the time to perform the log processing in the next iteration
will be less than or equal to the time that is specified for MAXRO. If this
condition is met, the next iteration is the last iteration.

• Db2 estimates that the SWITCH phase will not start by the deadline that is
specified for DEADLINE. If this condition is met, Db2 terminates reorganization.

• The number of log records that the next iteration is to process is not sufficiently
lower than the number of log records that were processed in the previous
iteration. If this condition is met but the first two conditions are not met, Db2
sends message DSNU377I to the console. Db2 continues log processing for the length of time that is specified for DELAY and then performs the action that is specified for LONGLOG.

**Operator actions**

LONGLOG specifies the action that Db2 performs if the pace of processing log records between iterations is slow. If no action is taken after message DSNU377I is sent to the console, the LONGLOG option automatically goes into effect. Some examples of possible actions that you can take:

- Execute the START DATABASE(database) SPACENAM(tablespace) ... ACCESS(RO) command and the QUIESCE utility to drain the write claim class. Db2 performs the last iteration, if MAXRO is not DEFER. After the QUIESCE, you should also execute the ALTER UTILITY command, even if you do not change any REORG parameters.
- Execute the START DATABASE(database) SPACENAM(tablespace) ... ACCESS(RO) command and the QUIESCE utility to drain the write claim class. Then, after reorganization makes some progress, execute the START DATABASE(database) SPACENAM(tablespace) ... ACCESS(RW) command. This increases the likelihood that processing of log records between iterations can continue at an acceptable rate. After the QUIESCE, you should also execute the ALTER UTILITY command, even if you do not change any REORG parameters.
- Execute the ALTER UTILITY command to change the value of MAXRO. Changing it to a huge positive value, such as 9999999, causes the next iteration to be the last iteration.
- Execute the ALTER UTILITY command to change the value of LONGLOG.
- Execute the TERM UTILITY command to terminate reorganization.
- Adjust the amount of buffer space that is allocated to reorganization and to applications. This adjustment can increase the likelihood that processing of log records between iterations can continue at an acceptable rate. After adjusting the space, you should also execute the ALTER UTILITY command, even if you do not change any REORG parameters.
- Adjust the scheduling priorities of reorganization and applications. This adjustment can increase the likelihood that processing of log records between iterations can continue at an acceptable rate. After adjusting the priorities, you should also execute the ALTER UTILITY command, even if you do not change any REORG parameters.

Db2 does not take the action specified in the LONGLOG phrase if any one of these events occurs before the delay expires:

- An ALTER UTILITY command is issued.
- A TERM UTILITY command is issued.
- Db2 estimates that the time to perform the next iteration is likely to be less than or equal to the time specified on the MAXRO keyword.
- REORG terminates for any reason (including the deadline).

**Related concepts:**

"Before running REORG TABLESPACE" on page 603

---

**Unloading without reloading**

REORG can unload data without continuing and without creating a SYSIBM.SYSUTIL record after the job ends.
If you specify UNLOAD ONLY, REORG unloads data from the table space and then ends. You can reload the data at a later date with the LOAD utility, specifying FORMAT UNLOAD.

Between unloading and reloading, you can add a validation routine to a table. During reloading, all the rows are checked by the validation procedure.

Do not use REORG UNLOAD ONLY to propagate data. When you specify the UNLOAD ONLY option, REORG unloads only the data that physically resides in the base table space; LOB and XML columns are not unloaded. For purposes of data propagation, you should use UNLOAD or REORG UNLOAD EXTERNAL instead.

REORG UNLOAD ONLY and REORG UNLOAD EXTERNAL cannot be used to unload inline LOBs.

**Reclaiming space from dropped tables**

Reorganization omits tables that were previously dropped, reclaiming the space that they acquired. For partition-by-growth table spaces, you cannot use REORG to reclaim the space.

**Related tasks:**

"Reclaiming space in the DBD" on page 395

---

**Reorganizing the catalog and directory**

You can run REORG TABLESPACE on the table spaces in the catalog database (DSNDB06) and on the SCT02, SPT01, DBD01, SYSLGRNX, SYSDBDXA, SYSSPUXA, and SYSSPUXB table spaces in the directory database (DSNDB01).

**Important:**

You must take a full image copy before and after reorganizing any catalog or directory object. Otherwise, you cannot recover any catalog or directory objects without the full image copies. When you reorganize the DSNDB06.SYSTSCPYP table space with the LOG NO option and omit the COPYDDN option, Db2 places the table space in COPY-pending status. Take a full image copy of the table space to remove the COPY-pending status before continuing to reorganize the catalog or directory table spaces.

Running REORG LOG NO COPYDDN avoids the COPY-pending status, because an inline copy is taken during the REORG.

The FASTSWITCH YES option is ignored for catalog and directory objects.

**When to run REORG on the catalog and directory**

You do not need to run REORG TABLESPACE on the catalog and directory table spaces as often as you do on user table spaces. RUNSTATS collects statistics about user table spaces, which you use to determine whether a REORG is necessary. You can use the same statistics to determine whether a REORG is needed for catalog and directory table spaces. However, Db2 does not generate statistics for certain items in the directory, such as the SYSLGRNX table space and its corresponding indexes DSNLXX01 and DSNLXX02.
Reorganize the whole catalog before a catalog migration or once every couple of years. Reorganizing the catalog is useful for reducing the size of the catalog table spaces. To improve query performance, reorganize the indexes on the catalog tables.

When statistical information indicates that you need to reorganize any of the catalog table spaces that are listed in the following table, you should also reorganize the corresponding directory table space. If the inline LOB length has changed, you should also reorganize any associated LOB directory table spaces.

Table 93. Catalog table spaces and their corresponding directory table spaces

<table>
<thead>
<tr>
<th>Catalog table space</th>
<th>Corresponding directory table space</th>
<th>Associated LOB directory table spaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNDB06.SYSTSFAN</td>
<td>DSNDB01.DBD01</td>
<td>DSNDB01.SYSDBDXA</td>
</tr>
<tr>
<td>DSNDB06.SYSTSFCOL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNDB06.SYSTSFLD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNDB06.SYSTSFOR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNDB06.SYSTSFIX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNDB06.SYSTSIPT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNDB06.SYSTSKEY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNDB06.SYSTSREL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNDB06.SYSTSSYN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNDB06.SYSTSTAUC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNDB06.SYSTSTICT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNDB06.SYSTSTAB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNDB06.SYSTSTISP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNDB06.SYSTSDBR</td>
<td>DSNDB01.SCT02</td>
<td>None</td>
</tr>
<tr>
<td>DSNDB06.SYSTSPLN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNDB06.SYSTSPLA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNDB06.SYSTSPLD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNDB06.SYSTSSTM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNDB06.SYSTSPKG</td>
<td>DSNDB01.SPT01</td>
<td>DSNDB01.SYSSPUXA</td>
</tr>
<tr>
<td>DSNDB06.SYSTSPKA</td>
<td>DSNDB01.SYSSPUXB</td>
<td></td>
</tr>
<tr>
<td>DSNDB06.SYSTSPKD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNDB06.SYSTSPKLD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNDB06.SYSTSPKLS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNDB06.SYSTSPKSS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNDB06.SYSTSPKSS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Associated directory table spaces**

When certain catalog table spaces are reorganized, you should also reorganize the associated directory table space. The associated directory table spaces are listed in the previous table.

**Limitations for reorganizing the catalog and directory**

- You cannot reorganize DSNDB01.SYSUTILX.
- If SHRLEVEL NONE is specified, the UNLOAD ONLY or UNLOAD EXTERNAL and LOG YES options are not allowed for catalog and directory table spaces. However, LOG YES is required if SHRLEVEL NONE is specified for the catalog LOB table spaces. If SHRLEVEL REFERENCE is specified, LOG NO must be specified.
- The SORTDEVT and SORTNUM options are ignored for the following catalog and directory table spaces:
  - DSNDB06.SYSTSFAN
  - DSNDB06.SYSTSFCOL

Chapter 28. REORG TABLESPACE  629
- DSNDB06.SYSTSFLD
- DSNDB06.SYSTSFOR
- DSNDB06.SYTSIXS
- DSNDB06.SYSTSIPT
- DSNDB06.SYTSKEY
- DSNDB06.SYTSREL
- DSNDB06.SYSTSSYN
- DSNDB06.SYSTSTAU
- DSNDB06.SYSTSTPT
- DSNDB06.SYSTSTAB
- DSNDB06.SYSTSTSP
- DSNDB06.SYSTDABA
- DSNDB06.SYSTDABU
- DSNDB06.SYSTDSTG
- DSNDB06.SYSTDOL
- DSNDB06.SYSTDVR
- DSNDB06.SYTSPLN
- DSNDB06.SYTSPLA
- DSNDB06.SYTSPLD
- DSNDB06.SYTSSTM
- DSNDB06.SYTSVWD
- DSNDB06.SYSTSVED
- DSNDB06.SYSTSVEW
- DSNDB06.SYTSVEW
- DSNDB06.SYTST
- DSNDB06.DBD01

- Any LOB table spaces, such as DSNDB01.SYSDBDXA (For more information about restricted REORG options for LOB table spaces, see “Reorganization of a LOB table space” on page 644).

The COPYDDN and RECOVERYDDN options are valid for the preceding catalog and directory tables if SHRLEVEL REFERENCE is also specified.

- REORG TABLESPACE with STATISTICS cannot collect inline statistics on the following catalog and directory table spaces:
  - DSNDB06.SYSTDFAU
  - DSNDB06.SYTSCOL
  - DSNDB06.SYSTSFOR
  - DSNDB06.SYSTSFLD
  - DSNDB06.SYSTSFAU
  - DSNDB06.SYSTSIPT
  - DSNDB06.SYTSKEY
  - DSNDB06.SYTSREL
  - DSNDB06.SYTSSTG
  - DSNDB06.SYTDABA
  - DSNDB06.SYTDABU
  - DSNDB06.SYTDVR
  - DSNDB06.SYTDVL
  - DSNDB06.SYTDLN
  - DSNDB06.SYTDPLA
  - DSNDB06.SYTDPLD
  - DSNDB06.SYTDSTM
  - DSNDB06.SYTDVWD
  - DSNDB06.SYTVWA
  - DSNDB06.SYTVW
  - DSNDB06.SYTSTATS
DSNDB06.SYSHIST
DSNDB01.DBD01
Any LOB table spaces, such as DSNDB01.SYSDBDXA (For more information about restricted REORG options for LOB table spaces, see “Reorganization of a LOB table space” on page 644.)

Phases for reorganizing the catalog and directory

REORG TABLESPACE processes certain catalog and directory table spaces differently from other table spaces; it does not execute the BUILD and SORT phases for the following table spaces:
- DSNDB06.SYSTSFAU
- DSNDB06.SYSTSCOL
- DSNDB06.SYSTSFLD
- DSNDB06.SYSTSFOR
- DSNDB06.SYSTSIXS
- DSNDB06.SYSTSIPT
- DSNDB06.SYSTSKEY
- DSNDB06.SYSTSREL
- DSNDB06.SYSTSSYN
- DSNDB06.SYSTSTAU
- DSNDB06.SYSTSTPT
- DSNDB06.SYSTSTAB
- DSNDB06.SYSTSTSP
- DSNDB06.SYSTSDBA
- DSNDB06.SYSTSDBU
- DSNDB06.SYSTSSTG
- DSNDB06.SYSTSVOL
- DSNDB06.SYSTSDBR
- DSNDB06.SYSTSPLN
- DSNDB06.SYSTSPLA
- DSNDB06.SYSTSPLD
- DSNDB06.SYSTSSTM
- DSNDB06.SYSTSVWD
- DSNDB06.SYSTSVEW
- DSNDB01.DBD01

For these table spaces, REORG TABLESPACE reloads the indexes (in addition to the table space) during the RELOAD phase, rather than storing the index keys in a work data set for sorting.

For all other catalog and directory table spaces, Db2 uses index build parallelism.

Changing data set definitions

If the table space is defined by storage groups, Db2 allocates space, and you cannot alter data set definitions while a REORG job is in process. Db2 deletes and redefines the necessary data sets to reorganize the object.

About this task

For REORG with SHRLEVEL REFERENCE or CHANGE, you can use the ALTER STOGROUP command to change the characteristics of a Db2-managed data set. To change the characteristics of a user-managed data set, specify the new characteristics when you create the shadow data set. For example, placing the original and shadow data sets on different disk volumes might reduce contention.
and improve the performance of REORG and the performance of applications
during REORG execution.

Related reference:
“Shadow data sets for REORG TABLESPACE” on page 614

Temporarily interrupting REORG

You can temporarily pause REORG TABLESPACE.

If you specify UNLOAD PAUSE, REORG pauses after unloading the table space
into the unload data set. You cannot use NOSYSREC and PAUSE. The job
completes with return code 4. You can restart REORG by using the phase restart or
current restart. Do not alter the REORG statement.

The REORG utility remains in stopped status until REORG is restarted or
terminated.

While REORG is interrupted by PAUSE, you can redefine the table space attributes
for user-defined table spaces. PAUSE is not required for STOGROUP-defined table
spaces. Attribute changes are done automatically by a REORG following an ALTER
TABLESPACE.

How to override dynamic sort work data set allocation

Db2 estimates how many records are to be sorted. This information is used for
dynamic allocation of sort work space. Sort work space is allocated by Db2 or by
the sort program that is used.

If the table space contains rows with VARCHAR columns, Db2 might not be able
to accurately estimate the number of records. If the estimated number of records is
too high, if the requested sort work space is not available, or if the estimated
number of records is too low, which causes the sort to overflow, the utility might
fail and cause an abend.

Recommendation: To enable Db2 to calculate a more accurate estimate:
• For a table space that is partitioned (non-universal), run RUNSTATS UPDATE
  ALL before REORG.
• For any other type of table space, run RUNSTATS UPDATE SPACE before
  REORG.

When you run RUNSTATS with SHRLEVEL REFERENCE, real-time statistics
values are also updated.

You can override the dynamic allocation of sort work space in one of the following
ways:
• Allocate the sort work data sets with SORTWKnn DD statements in your JCL.
• If the number of rows in the affected table space in column TOTALROWS of
table SYSIBM.SYSTABLESPACESTATS is not available or is significantly
incorrect, you can update the value to a more appropriate value using an SQL
  UPDATE statement. When REORG on the affected table space completes,
TOTALROWS is set to the number of rows in the associated table space.
• If the number of keys for an associated index in column TOTALENTRIES of
table SYSIBM.SYSINDEXSPACESTATS is not available or is significantly
incorrect, you can update the value to a more appropriate value using an SQL
UPDATE statement. The next time that REBUILD INDEX is run, TOTALENTRIES is set to the number of keys for the affected index.

Related reference:
- Db2 Sort for z/OS

Related information:
- DFSORT Application Programming Guide

Redistributing data across partitions by using REORG

When data becomes skewed across partitions performance can be slower. You can correct the problem by redistributing the data more evenly across partitions. One way to redistribute the data is to let the REORG TABLESPACE utility determine any limit key changes and redistribute the data accordingly.

About this task

Alternatively, you can explicitly specify limit key values. If you want to specify your own limit key values, follow the instructions in Changing the boundary between partitions (Db2 Administration Guide).

Procedure

To redistribute data across partitions by using REORG:

Run the REORG TABLESPACE utility with the REBALANCE option. REBALANCE specifies that you want Db2 to determine the limit key changes for the partitioned table space and redistribute the data accordingly. The data remains available.

Restriction: REBALANCE is not allowed in any of the following situations:
- With the SCOPE PENDING option
- For partitioned-by-growth table spaces
- For table spaces with pending limit key changes

See the description of REBALANCE in the description of the REORG TABLESPACE syntax for a complete list of restrictions.

If the table has a clustering index that does not match the partitioning key, you must run REORG TABLESPACE twice. Running REORG twice ensures that the data is rebalanced and all rows are in clustering order. The first utility execution rebalances the data and the second utility execution sorts the data.

For example, assume that you have a table space that was created with the following SQL:

```
CREATE TABLESPACE TS IN DB
    USING STOGROUP SG
    NUMPARTS 4 BUFFERPOOL BP0;
CREATE TABLE TB (C01 CHAR(5) NOT NULL,
    C02 CHAR(5) NOT NULL,
    C03 CHAR(5) NOT NULL)
    IN DB.TS
    PARTITION BY (C01)
    (PART 1 VALUES ('00001'),
```
To rebalance the data across the four partitions, use the following REORG TABLESPACE control statement:

```
REORG TABLESPACE DB.TS REBALANCE
```

After this utility job completes, the table space is placed in advisory REORG-pending (AREO*) status to indicate that a subsequent reorganization is recommended to ensure that the rows are in clustering order. For this subsequent reorganization, use the following REORG TABLESPACE control statement:

```
REORG TABLESPACE DB.TS
```

**Related reference:**

“Syntax and options of the REORG TABLESPACE control statement” on page 556

Appendix B, “Advisory or restrictive states,” on page 1037

---

### How partitions can be unloaded and reloaded in parallel

In some situations, the REORG utility attempts to unload and reload partitions in parallel. In other situations, parallel unloading and reloading does not occur.

REORG attempts to unload and reload table space partitions in parallel in the following situations:

- If you specify the NOSYSREC keyword.
- If the NOSYSREC keyword is defaulted like it is for SHRLEVEL CHANGE.
- If you specify the UNLDDN keyword with a template name, where the template's data set name pattern includes a partition number.

REORG does not attempt to unload and reload table space partitions in parallel in the following situations:

- If the DATAWKnn DD statements are coded in the JCL.
- If you do not specify the SORTDEVT keyword.
- If the UTPRINT data set is not allocated to SYSOUT.
- If you specify the REBALANCE keyword.
- If rows might move from one partition to another as a result of alter limit keys or partition-by-growth table space.
- If you specify the UNLDDN keyword with a template name and specify UNIT(TAPE) STACK(YES).).
- When the number of subtasks that are started in parallel exceeds the value of the PARALLEL option or the PARAMDEG_UTIL subsystem parameter.

### How to use inline copy with REORG TABLESPACE

You can create a full image copy data set (SHRLEVEL REFERENCE) during REORG TABLESPACE execution.

The new copy is an inline copy. The advantage to using an inline copy is that the table space is not left in COPY-pending status, regardless of which LOG option is specified for the utility. Thus, data availability is increased. You must take an inline copy when you specify the REBALANCE option.
To create an inline copy, use the COPYDDN and RECOVERYDDN keywords. You can specify up to two primary copies and two secondary copies. Inline copies are produced during the RELOAD phase of REORG processing.

You can specify a template name for COPYDDN or RECOVERYDDN. If that template uses the &PA. or &PART. variable, REORG allocates as many copy data sets as the number of partitions that are being reorganized.

The SYSCOPY record that is produced by an inline copy contains ICTYPE=F, SHRLEVEL=R. The STYPE column contains an X if the image copy was produced by REORG TABLESPACE LOG(YES), and a W if the image copy was produced by REORG TABLESPACE LOG(NO). The data set that is produced by the inline copy is logically equivalent to a full image copy with SHRLEVEL REFERENCE, but the data within the data set differs in some respects:

- Data pages might be out of sequence and some might be repeated. If pages are repeated, the last one is always the correct copy.
- Space map pages are out of sequence and might be repeated

The total number of duplicate pages is small, with a negligible effect on the amount of space that is required for the data set. One exception to this guideline is the case of running REORG SHRLEVEL CHANGE, in which the number of duplicate pages varies with the number of records that are applied during the LOG phase.

**Related reference:**
Chapter 34, “TEMPLATE,” on page 793

---

**Creating a FlashCopy image copy with REORG TABLESPACE**

As part of REORG TABLESPACE processing, you can use FlashCopy technology to take image copies. This method is potentially faster than the traditional Db2 utility methods for creating inline copies and thus reduces the time that data is unavailable. FlashCopy image copies can also potentially reduce the time that is required for recovery operations.

**About this task**

REORG TABLESPACE can also create one to four additional inline image copies by using the traditional methods. Traditional inline image copies are output to a non-VSAM sequential format data set. For more information about traditional inline copies, see “How to use inline copy with REORG TABLESPACE” on page 634.

**Restriction:** You cannot create FlashCopy image copies if you specify UNLOAD ONLY or UNLOAD EXTERNAL in the REORG TABLESPACE utility control statement.

**Procedure**

To create a FlashCopy image copy with REORG TABLESPACE:

Specify FLASHCOPY(YES) or FLASHCOPY(CONSISTENT) in the REORG TABLESPACE utility control statement. Alternatively, you can set the FLASHCOPY_REORG_TS subsystem parameter to YES, which specifies that REORG TABLESPACE is to use FLASHCOPY(YES) by default. The value that you specify for the FLASHCOPY option in the REORG TABLESPACE statement always
overrides the value for the FLASHCOPY_REORG_TS subsystem parameter. Optionally, you can also specify FCCOPYDDN in the REORG TABLESPACE statement. Use this option to specify a template for the FlashCopy image copy. If you do not specify the FCCOPYDDN option in the REORG TABLESPACE statement, the utility uses the value from the FCCOPYDDN subsystem parameter.

**Restriction:** The data sets that you specify for the FlashCopy image copy must be on FlashCopy Version 2 disk volumes. When you specify FLASHCOPY(YES) or FLASHCOPY(CONSISTENT), REORG TABLESPACE uses FlashCopy technology to create a consistent copy of the target objects. If you also requested one or more traditional inline copies in the REORG TABLESPACE statement (by specifying COPYDDN or RECOVERYDDN), the utility also creates those copies. REORG TABLESPACE does not use the FlashCopy image copy to create those traditional inline copies.

When you request a FlashCopy image copy, but you do not specify the COPYDDN option in the REORG TABLESPACE option, and you do not include a SYSCOPY DD statement or a TEMPLATE statement with a SYSCOPY data set specification, REORG TABLESPACE does not create an inline image copy as well as a FlashCopy image copy. The table space is not placed in the COPY-pending state. However, when you request a FlashCopy image copy, and you do not specify the COPYDDN option in a REORG TABLESPACE statement, but you include a SYSCOPY DD statement or a TEMPLATE statement with a SYSCOPY data set specification, REORG TABLESPACE creates an inline image copy as well as a FlashCopy image copy.

**Important:** You should request a sequential image copy as well as a FlashCopy image copy when either of the following conditions are true:

- Your environment and system setup for FlashCopy image copies is not yet stable and predictable. If the FlashCopy process fails during the SWITCH phase, COPY-pending status is set if a sequential image copy is not taken.
- You want the FlashCopy image copy for fast local recovery, but require a sequential image copy to be shipped to a remote site for disaster recovery.

Related objects are copied if one of the following conditions is true:

- You specified REBALANCE in the REORG TABLESPACE statement.
- The partitioning key has changed since the last time the table space was reorganized
- The base table space is a partition-by-growth table space.

Failures occur in the following situations:

- The FlashCopy image copy fails if the FlashCopy Version 2 disk volumes are not available or if any of the other FlashCopy operational restrictions exist. For a list of those operational restrictions, see “FlashCopy image copies” on page 157.
- REORG TABLESPACE terminates if the FlashCopy image copy for the target table space fails, you specified SHRLEVEL REFERENCE or SHRLEVEL CHANGE, and inline copies were not taken. (If the FlashCopy image copy fails and you specified SHRLEVEL REFERENCE or SHRLEVEL CHANGE, but inline copies were taken, the utility continues.)

**Related concepts:**

“FlashCopy image copies” on page 157

**Related reference:**

- DEFAULT TEMPLATE field (FCCOPYDDN subsystem parameter) (Db2 Installation and Migration)
Improving REORG TABLESPACE performance

You can improve the performance of the REORG TABLESPACE utility by taking certain actions.

About this task

**Recommendation:** Run online REORG during light periods of activity on the table space or index.

Procedure

To improve REORG TABLESPACE performance:

- Run REORG concurrently on separate partitions of a partitioned table space if no nonpartitioned indexes exist. When you run REORG on partitions of a partitioned table space, the sum of each job's processor usage is greater than for a single REORG job on the entire table space. However, the elapsed time of reorganizing the entire table in parallel can be significantly less than it would be for a single REORG job.
- Use parallel index build for table spaces or partitions that have more than one defined index.
- Specify NOSYSREC on your REORG statement.
- If you are not using NOSYSREC, use an UNLDDN template to enable unload parallelism.
- If you are using 3990 caching, and you have the nonpartitioning indexes on RAMAC, consider specifying YES on the UTILaney CACHE OPTION field of installation panel DSNTIPE.

This option allows Db2 to use sequential prestaging when reading data from RAMAC for the following utilities:
- LOAD PART integer RESUME
- REORG TABLESPACE PART

For LOAD PART and REORG TABLESPACE PART utility jobs, prefetch reads remain in the cache longer, which can lead to possible improvements in the performance of subsequent writes.

For REORG with SHRLEVEL CHANGE or SHRLEVEL REFERENCE, use inline statistics only if you can afford the additional cost of collecting statistics inline. Collecting statistics inline makes it unnecessary to run another RUNSTATS job after the REORG job. However, collecting statistics inline might substantially increase the length of time that the REORG job runs.

- Schedule REORG with SHRLEVEL CHANGE when the rate of writing is low and transactions are short. Avoid scheduling REORG with SHRLEVEL CHANGE when critical applications are executing.

Under certain circumstances, the log records that REORG SHRLEVEL CHANGE uses contain additional information, as if DATA CAPTURE CHANGES were used. Generation of the additional information can slow applications and increase consumption of log space. The additional information is generated for all the tables in the table space if at least one table satisfies all these conditions:
- The table has undergone ALTER TABLE ADD column.
- The table does not use DATA CAPTURE CHANGES.
One of these conditions is true:

- The area that is being reorganized uses data compression.
- The area is a partitioned table space, and at least one partition uses data compression.

- Run REORG with DRAIN_WAIT.

  The DRAIN_WAIT option gives you greater control over the time that online REORG is to wait for drains. Also because the DRAIN_WAIT is the aggregate time that online REORG is to wait to perform a drain on a table space and associated indexes, the length of drains is more predictable than if each partition and index has its own individual waiting time limit.

  By specifying a short delay time (less than the system timeout value, IRLMRWT), you can reduce the impact on applications by reducing timeouts. You can use the RETRY option to provide more opportunities for the online REORG to complete successfully. If you do not want to use RETRY processing, you can still use DRAIN_WAIT to set a specific and more consistent limit on the length of drains.

  RETRY allows an online REORG that is unable to drain the objects that it requires so that Db2 can try again after a set period (RETRY_DELAY). During the RETRY_DELAY period, all the objects are available for read-write access in the case of SHRLEVEL CHANGE. For SHRLEVEL REFERENCE, the objects remain with the access that existed prior to the attempted drain (that is if the drain fails in the UNLOAD phase the object remains in read-write access; if the drain fails in the SWITCH phase, objects remain in read-only access).

  Because application SQL statements can queue behind any unsuccessful drain that the online REORG has tried, define a reasonable delay before you try again to allow this work to complete; the default is 5 minutes.

  When you specify DRAIN_WRITERS (the default) with SHRLEVEL CHANGE and RETRY, multiple read-only log iterations can occur. Generally, online REORG might need to do more work when RETRY is specified, and this might result in multiple or extended periods of restricted access. Applications that run alongside online REORG need to perform frequent commits. During the interval between retries, the utility is still active, and consequently other utility activity against the table space and indexes is restricted.

- Run the REORG TABLESPACE utility with the PART SHRLEVEL REFERENCE or PART SHRLEVEL CHANGE option specified and the SORTNPSI YES or SORTNPSI AUTO option specified or subsystem parameter REORG_PART_SORT_NPSI enabled. When you run REORG TABLESPACE with these options, REORG TABLESPACE sorts all keys of the nonpartitioned secondary indexes and builds the shadow index from the sorted keys.

### Parallel index building for REORG TABLESPACE

Parallel index building reduces the elapsed time for a REORG TABLESPACE job by sorting the index keys and rebuilding multiple indexes in parallel, rather than sequentially. Optimally, a pair of subtasks processes each index; one subtask sorts extracted keys, whereas the other subtask builds the index.

REORG TABLESPACE begins building each index as soon as the corresponding sort emits its first sorted record. The following figure shows the flow of a REORG TABLESPACE job that uses a parallel index build. Db2 starts multiple subtasks to sort index keys and build indexes in parallel. If you specify STATISTICS, additional subtasks collect the sorted keys and update the catalog table in parallel, eliminating the need for a second scan of the index by a separate RUNSTATS job.
REORG TABLESPACE uses parallel index build if more than one index needs to be built (including the mapping index for SHRLEVEL CHANGE). You can either let the utility dynamically allocate the data sets that SORT needs for this parallel index build or provide the necessary data sets yourself. The number of subtasks must be less than or equal to the number that is specified by the PARALLEL option. If you do not specify the PARALLEL option, the PARAMDEG_UTIL subsystem parameter determines the maximum degree of parallelism for the utility.

Select one of the following methods to allocate sort work and message data sets:

**Method 1:**

REORG TABLESPACE determines the optimal number of sort work data sets and message data sets.

1. Specify the SORTDEVT keyword in the utility statement.
2. Allow dynamic allocation of sort work data sets by not supplying SORTWKnn DD statements in the REORG TABLESPACE utility JCL.
3. Allocate UTPRINT to SYSOUT.

**Method 2:**

Control allocation of sort work data sets, while REORG TABLESPACE allocates message data sets.

1. Provide DD statements with DD names in the form SWnnWKmm.
2. Allocate UTPRINT to SYSOUT.

**Method 3:**

Exercise the most control over rebuild processing; specify both sort work data sets and message data sets.

1. Provide DD statements with DD names in the form SWnnWKmm.
2. Provide DD statements with DD names in the form UTPRINnn.
Data sets used

If you select Method 2 or 3 in the preceding information, define the necessary data sets by using the following information.

Each sort subtask must have its own group of sort work data sets and its own print message data set. Possible reasons to allocate data sets in the utility job JCL rather than using dynamic allocation are:

- To control the size and placement of the data sets
- To minimize device contention
- To optimally use free disk space
- To limit the number of utility subtasks that are used to build indexes

The DD name SWnnWKmm defines the sort work data sets that are used during utility processing. nn identifies the subtask pair, and mm identifies one or more data sets that are to be used by that subtask pair. For example:

**SW01WK01**

- Is the first sort work data set that is used by the subtask that builds the first index.

**SW01WK02**

- Is the second sort work data set that is used by the subtask that builds the first index.

**SW02WK01**

- Is the first sort work data set that is used by the subtask that builds the second index.

**SW02WK02**

- Is the second sort work data set that is used by the subtask that builds the second index.

The DD name UTPRINnn defines the sort work message data sets that are used by the utility subtask pairs. nn identifies the subtask pair.

Every time you invoke REORG TABLESPACE, new UTPRINnn data sets are dynamically allocated. REORG TABLESPACE does not reuse UTPRINnn data sets from previous job steps. This behavior might cause the available JES2 job queue elements to be consumed more quickly than expected.

Number of sort subtasks

The maximum number of utility subtask pairs that are started for parallel index build is equal to the number of indexes that need to be built.

REORG TABLESPACE determines the number of subtask pairs according to the following guidelines:

- The number of subtask pairs equals the number of allocated sort work data set groups.
- The number of subtask pairs equals the number of allocated message data sets.
- If you allocate both sort work data sets and message data set groups, the number of subtask pairs equals the smallest number of allocated data sets.
Allocation of sort subtasks

REORG TABLESPACE attempts to assign one sort subtask pair for each index that is to be built. If REORG TABLESPACE cannot start enough subtasks to build one index per subtask pair, it allocates any excess indexes across the pairs; therefore one or more subtask pairs might build more than one index.

During parallel index build processing, REORG distributes all indexes among the subtask pairs according to the index creation date, assigning the first created index to the first subtask pair. For SHRLEVEL CHANGE, the mapping index is assigned last.

Estimating the sort work file size

If you choose to provide the data sets, you need to know the size and number of keys that are present in all of the indexes that are being processed by the subtask in order to calculate each sort work file size. After you determine which indexes are assigned to which subtask pairs, use the following formula to calculate the required space:

\[ 2 \times (\text{longest index key} + c) \times \text{(number of extracted keys)} \]

**longest key**

The length of the longest index key that is to be processed by the subtask. If the index is of varying length, the longest key is the maximum possible length of a key with all varying-length columns that are padded to their maximum length, plus 2 bytes for each varying-length column in the index. For example, if an index with three columns (A, B, and C) has length values of CHAR(8) for A, VARCHAR(128) for B, and VARCHAR(50) for C, the longest key is calculated as follows:

\[ 8 + 128 + 50 + 2 + 2 = 190 \]

For SHRLEVEL CHANGE, the mapping index key length is 21.

**c**

A value as follows:

- 10 if the indexes that are rebuilt are a mix of data-partitioned secondary indexes and nonpartitioned indexes
- 8 if all indexes are partitioned or none of them are data-partitioned secondary indexes.

**number of extracted keys**

The number of keys from all indexes that need to be sorted and that are to be processed by the subtask.

When you calculate the sort work data set size, do not count keys that are not sorted. Keys are not sorted when both of the following conditions are true:

- SORTDATA is in effect for REORG TABLESPACE, and the keys belong to a partitioning, clustering index.
- The table space is a partitioned table space, and data partitions are not being unloaded and reloaded in parallel.

The space estimation formula might indicate that 0 bytes are required, because the only index that is processed is the partitioning, clustering index. In this case, if you allocate your own groups of sort work data sets, you still need to allocate sort work data sets, but you can use a minimal allocation, such as 1 track.

Related tasks:
How Db2 unloads data

Db2 unloads data by table space scan with sort, table space scan, or clustering index.

Db2 unloads data by one of three methods:

- **Table space scan with sort**: If at least one table space has an index, Db2 uses a table space scan with a sort.
- **Table space scan**: Db2 uses a table space scan for simple table spaces that contain more than one table, or that contain one table but do not have an index.
- **Clustering index**: Db2 uses this option for simple table spaces that contain one table and have an index, and for tables in a segmented table space that have an index.

Failure during the RELOAD phase

Failure during the RELOAD phase (after the data is unloaded and data sets are deleted, but before the data is reloaded) results in an unusable table space.

If the error is on the table space data:

- If you have defined data sets, you can allocate new data sets.
- If STOGROUP has defined data sets, you can alter the new table space to change the primary and secondary quantities.
- If you allocate new data sets, alter the table space, or add volumes to the storage group, restart the REORG job at the beginning of the phase. Otherwise, you can restart either at the last commit point or at the beginning of the phase.

If the error is on the unloaded data, or if you used the NOSYSREC option, terminate REORG by using the **TERM UTILITY** command. Then recover the table space, using **RECOVER**, and run the REORG job again.

Reorganization of partitioned table spaces

If you reorganize a single partition or a range of partitions, all indexes of the table space are affected. Depending on how disorganized the nonpartitioning indexes are, you might want to reorganize them.

**Related tasks**:

- “Determining which indexes require reorganization” on page 541

Reorganization of partition-by-growth table spaces

When you run the **REORG TABLESPACE** utility on a partition-by-growth table space, the utility condenses the data into the fewest partitions possible. If necessary, the utility adds partitions, if possible.

If you reorganize a partition-by-growth table space that does not contain LOB columns, **REORG TABLESPACE** condenses the data into the minimum number of required partitions. Because the utility cannot reclaim physical space, the excess partitions are empty. If the data needs additional space and the maximum number of partitions has not been reached, **REORG TABLESPACE** adds additional partitions. If the maximum number of partitions has been reached, the utility fails.
If the partition-by-growth table space contains LOB columns, REORG TABLESPACE minimizes partitions by eliminating existing holes, but does not move the data from one partition to another.

When you reorganize a partition-by-growth table space at the partition level, REORG TABLESPACE minimizes partitions by eliminating existing holes. If you specify REORG TABLESPACE PART for a partition-by-growth table space and the data does not fit back into its partition, the utility fails. This situation can happen for one of the following reasons:

- A partition does not have enough space to insert the data and system page for the table.
  
  To prevent the utility from failing, run REORG TABLESPACE on the entire table space.

- The amount of free space changes during the REORG.
  
  To prevent the utility from failing, run REORG TABLESPACE on the entire table space, or modify the PCTFREE parameter to fit the data rows into the partition.

In the following situations, the addition of new partitions might lead to the failure of REORG TABLESPACE because of a lack of disk space:

- When REORG TABLESPACE is run against a subset of the partitions in a partition-by-growth table space
- When REORG TABLESPACE AUX NO is run against a partition-by-growth table space in which a table contains LOB columns

For these situations only, you can alleviate space problems by taking one of the following actions:

- Execute ALTER TABLESPACE on the table space to change PCTFREE and FREEPAGE to 0. This action is the preferred solution.
- Instead of running REORG TABLESPACE against a subset of the partitions in a partition-by-growth table space, run REORG on the entire table space.
- Set the REORG_IGNOR FREESPACE subsystem parameter to YES. Doing so causes REORG TABLESPACE to not honor the PCTFREE and FREEPAGE values that are defined for the table space. REORG uses 0 for the PCTFREE and FREEPAGE values when it reloads data into the table space.

REORG parallelism does not apply to partition-by-growth table spaces. This restriction ensures that REORG TABLESPACE is able to condense the data into the minimum number of required partitions.

If a compression dictionary exists, the compression dictionary is copied to all partitions even if the partition is empty.

Related tasks:

- Reserving free space for table spaces (Db2 Performance)

Related reference:

- Chapter 28, “REORG TABLESPACE,” on page 553
- ALTER TABLESPACE (Db2 SQL)
- REORG IGNORE FREESPACE field (REORG_IGNOR FREESPACE subsystem parameter) (Db2 Installation and Migration)
Reorganization of segmented table spaces

The REORG TABLESPACE utility reorganizes segmented table spaces.

If the target table space is segmented, REORG unloads and reloads by table.

If an index exists on a table in a segmented table space, that table is unloaded in clustering sequence. If NO index exists, the table is unloaded in physical row and segment order.

For segmented table spaces, REORG does not normally need to reclaim space from dropped tables. Space that is freed by dropping tables in a segmented table space is immediately available if the table space can be accessed when DROP TABLE is executed. If the table space cannot be accessed when DROP TABLE is executed (for example, the disk device is offline), Db2 removes the table from the catalog, but does not delete all table rows. In this case, the space for the dropped table is not available until REORG reclaims it.

After you run REORG, the segments for each table are contiguous.

Comparison of the numbers of loaded and unloaded records

At the end of the RELOAD phase, REORG compares the number of records that were actually loaded to the number of records that were unloaded.

If the counts do not match, the resulting actions depend on the UNLOAD option that you specified on the original job:

• If you specify UNLOAD PAUSE, REORG sets return code 4 and continues processing the job.
• If you specify UNLOAD CONTINUE, Db2 issues an error message and abnormally terminates the job. The table space or partition remains in RECOVER-pending status.

Reorganization of a LOB table space

Run the REORG TABLESPACE utility on a LOB table space to help increase the effectiveness of prefetch processing. When you run REORG on a LOB table space, the utility removes embedded free space, attempts to make LOB pages contiguous, and reclaims physical space.

You can reorganize a LOB table space separately from the base table space or you can reorganize the base and LOB table spaces together. To reorganize them together, specify the base table space name in the REORG statement and the AUX YES option. For restrictions, see [AUX YES].

To reorganize a LOB table space separately from the base table space, specify the LOB table space name in the REORG statement. REORG unloads LOBs to a shadow data set and reclaims any physical space.

For LOB table spaces, certain REORG statement options are not applicable, including the following options:

• AUTOESTSPACE
• AUX YES
• DISCARD
When you reorganize a LOB table space, you cannot specify SHRLEVEL NONE. You must specify SHRLEVEL REFERENCE or CHANGE. You must also specify LOG NO and take an inline image copy.

REORG SHRLEVEL CHANGE processes a LOB table space the same as REORG SHRLEVEL REFERENCE except that the mapping table is ignored. REORG SHRLEVEL CHANGE uses shadow data sets and includes a LOG phase.

Reorganization of an XML table space

Reorganizing an XML table space is a separate task from reorganizing the base table space.

When you specify the name of the base table space in the REORG statement, Db2 reorganizes only that table space and not any related XML objects. If you want Db2 to reorganize the XML objects, you must specify those object names. When you specify that you want XML table spaces to be reorganized, you must also specify the WORKDDN keyword and provide the specified temporary work file. The default is SYSUT1.

When you run REORG on an XML table space that supports XML versions, REORG discards rows for versions of an XML document that are no longer needed.

For XML table spaces and base table spaces with XML columns, you cannot specify the following options in the REORG statement:

- DISCARD
- REBALANCE
- **UNLOAD EXTERNAL**

In the following example, the REORG statement specifies that Db2 is reorganizing table space BASETS01 and XML table spaces XML1TS01 and XML2TS01. During this reorganization Db2 is to take an inline copy of the base table space and gather statistics for all three table spaces.

```sql
//STEP1  EXEC DSNUPROC,UID='HUHRU252.REORG1',TIME=1440,
  //  UTPROC='',
  //  SYSTEM='DSN',DB2LEV=DB2A
//SYSREC  DD DSN=HUHRU252.REORG1.STEP1.SYSREC,DISP=(MOD,DELETE,CATLG),
  //  UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSCOPY1 DD DSN=HUHRU252.REORG1.STEP1.SYSCOPY1,
  //  DISP=(MOD,CATLG),UNIT=SYSDA,
  //  SPACE=(4000,(20,20),,,ROUND)
//SYSUT1 DD DSN=IUJLU101.REORG.STEP1.SYSUT1,DISP=(MOD,DELETE,CATLG),
  //  UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSUT2 DD DSN=IUJLU101.REORG.STEP1.SYSUT2,DISP=(MOD,DELETE,CATLG),
  //  UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)

//SYSIN  DD *
  REORG TABLESPACE DBHR5201.BASETS01
    SHRLEVEL CHANGE MAPPINGTABLE MAP1
    COPYDDN(SYSCOPY1)
    STATISTICS TABLE(ALL)
    INDEX(ALL)
  REORG TABLESPACE DBHR5201.XML1TS01
    SHRLEVEL CHANGE MAPPINGTABLE MAP2
    STATISTICS TABLE(ALL)
    INDEX(ALL)
    WORKDDN(SYSUT1)
  REORG TABLESPACE DBHR5201.XML2TS01
    SHRLEVEL CHANGE MAPPINGTABLE MAP3
    STATISTICS TABLE(ALL)
    INDEX(ALL)
    WORKDDN(SYSUT2)
/*

---

**Reorganization with pending definition changes**

A REORG utility operation that materializes pending definition changes is equivalent to a data definition change that happens during an online transaction. REORG requires the same type of serialization that any data definition change requires in addition to the existing draining serialization it performs on the target object.

Therefore, to avoid possible application outages, try to run REORG at a time when the data is not heavily accessed, which allows for the data definition change to be materialized.

A REORG operation that materializes definition changes enables you to alter certain objects without significantly impacting availability. The data is unavailable for only a short time. This short time of unavailability happens during any REORG SHRLEVEL CHANGE operation.

However, a REORG operation that materializes pending definition changes does have a cost. REORG begins serialization with the concurrent application during the last log iteration in the LOG phase and changes schema definitions during the SWITCH phase. During this time, plans, packages, and the dynamic statement cache are quiesced and invalidated, and statistics and dynamic SQL are blocked. As a result, the DRAIN ALL behavior that REORG performs when it does not
materialize pending definition changes is not sufficient. For example, assume that an agent sees a segmented table space and claims and waits for REORG to release the drain. At the same time, REORG materializes the pending definition change and changes the table space from segmented to partition-by-growth. If REORG does not block the agent before its claim, after REORG releases the drain, the agent continues to access the table space with the attributes that it had before REORG ran. This situation can result in unpredictable behaviors. Application lock timeout errors are possible, because REORG was unable to acquire the locks on either the SYSIBM.SYSTABLESPACE record or on the plans or packages. The IRLM lock timeout value applies for these locks. REORG can hold these locks for longer than the IRLMRWT threshold because it must acquire multiple locks. (When REORG is not materializing definition changes, it needs only a single drain lock.) The value of the DRAIN_WAIT utility option does not apply to these lock timeouts.

Related concepts:
“Access with REORG TABLESPACE SHRLEVEL” on page 624

Related tasks:
Materializing pending definition changes (Db2 Administration Guide)

Related reference:
[RESOURCE TIMEOUT field (IRLMRWT subsystem parameter) (Db2 Installation and Migration)]

Compression after materialization of inline LOB changes

The REORG TABLESPACE utility can be run on a base table space to materialize the conversion of an auxiliary LOB table space to an inline LOB. However, if the object is defined with compression, the result of the reorganization can be little or no row compression in some of the partitions.

Introductory concepts
The effect of data compression on performance (Introduction to Db2 for z/OS)

To achieve better compression, run REORG TABLESPACE a second time on the table space that contains the inline LOB to build a new compression dictionary and attempt better compression.

Termination of REORG TABLESPACE

You can terminate the REORG TABLESPACE utility.

If you terminate REORG TABLESPACE with the TERM UTILITY command during the UNLOAD phase, objects have not yet been changed, and you can rerun the job.

If you terminate REORG TABLESPACE with the TERM UTILITY command during the RELOAD phase, the behavior depends on the SHRLEVEL option:

- For SHRLEVEL NONE, the data records are not erased. The table space and indexes remain in RECOVER-pending status. After you recover the table space, rerun the REORG job.
- For SHRLEVEL REFERENCE or CHANGE, the data records are reloaded into shadow objects, so the original objects have not been affected by REORG. You can rerun the job.
If you terminate REORG with the **TERM UTILITY** command during the SORT, BUILD, or LOG phases, the behavior depends on the SHRLEVEL option:

- For SHRLEVEL NONE, the indexes that are not yet built remain in RECOVER-pending status. You can run REORG with the SORTDATA option, or you can run REBUILD INDEX to rebuild those indexes.
- For SHRLEVEL REFERENCE or CHANGE, the records are reloaded into shadow objects, so the original objects have not been affected by REORG. You can rerun the job.

If you terminate a stopped REORG utility with the **TERM UTILITY** command during the SWITCH phase, the following conditions apply:

- All data sets that were renamed to their shadow counterparts are renamed to their original names, so that the objects remain in their original state, and you can rerun the job.
- If a problem occurs in renaming the data sets to the original names, the objects remain in RECOVER-pending status, and you cannot rerun the job.

If the SWITCH phase does not complete, the image copy that REORG created is not available for use by the RECOVER utility. If you terminate an active REORG utility during the SWITCH phase with the **TERM UTILITY** command, during the rename process, the renaming occurs, and the SWITCH phase completes. The image copy that REORG created is available for use by the RECOVER utility.

The REORG-pending status is not reset until the UTILTERM execution phase. If the REORG utility abnormally terminates or is terminated, the objects remain in REORG-pending status and RECOVER-pending status, depending on the phase in which the failure occurred.

The following table lists the restrictive states that REORG TABLESPACE sets according to the phase in which the utility terminated.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Effect on restrictive status</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNLOAD</td>
<td>No effect.</td>
</tr>
<tr>
<td>RELOAD</td>
<td>SHRLEVEL NONE:</td>
</tr>
<tr>
<td></td>
<td>• Places table space in RECOVER-pending status at the beginning of the phase and resets the status at the end of the phase.</td>
</tr>
<tr>
<td></td>
<td>• Places indexes in RECOVER-pending status.</td>
</tr>
<tr>
<td></td>
<td>• Places the table space in COPY-pending status. If COPYDDN is specified and SORTKEYS is ignored, the COPY-pending status is reset at the end of the phase. SORTKEYS is ignored for several catalog and directory table spaces</td>
</tr>
<tr>
<td></td>
<td>SHRLEVEL REFERENCE or CHANGE has no effect.</td>
</tr>
<tr>
<td>SORT</td>
<td>No effect.</td>
</tr>
<tr>
<td>BUILD</td>
<td>SHRLEVEL NONE resets RECOVER-pending status for indexes and, if the utility job includes both COPYDDN and SORTKEYS, resets COPY-pending status for table spaces at the end of the phase. SHRLEVEL REFERENCE or CHANGE has no effect.</td>
</tr>
<tr>
<td>SORTBLD</td>
<td>No effect during the sort portion of the SORTBLD phase. During the build portion of the SORTBLD phase, the effect is the same as for the BUILD phase.</td>
</tr>
<tr>
<td>LOG</td>
<td>No effect.</td>
</tr>
</tbody>
</table>
Table 94. Restrictive states that REORG TABLESPACE sets. (continued)

<table>
<thead>
<tr>
<th>Phase</th>
<th>Effect on restrictive status</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWITCH</td>
<td>No effect. Under certain conditions, if TERM UTILITY is issued, it must complete successfully; otherwise, objects might be placed in RECOVER-pending status.</td>
</tr>
</tbody>
</table>

Recovering a failed REORG job

If you terminate REORG SHRLEVEL NONE in the RELOAD phase, all SYSLGRNX records associated with the reorganization are deleted. Use the RECOVER TABLESPACE utility to recover to the current point in time. This action recovers the table space to its state before the failed reorganization.

Related concepts:
“Reorganizing the catalog and directory” on page 628

Related tasks:
“Terminating an online utility” on page 43

Related reference:
Appendix B, “Advisory or restrictive states,” on page 1037

Restart of REORG TABLESPACE

You can restart a REORG TABLESPACE utility job.

By default, Db2 uses RESTART(CURRENT) when restarting REORG TABLESPACE jobs, with the following exceptions:

- Jobs that are restarted in the SORT, BUILD, or SWITCH phase use RESTART(PHASE) by default.
- Jobs with the SORTKEYS option that are restarted in the RELOAD, SORT, BUILD, or SORTBILD phase always restart from the beginning of the RELOAD phase.
- Jobs with the SHRLEVEL REFERENCE, NOSYSREC, and SORTDATA options use RESTART(PHASE) to restart at the beginning of the UNLOAD phase.
- Jobs with unload parallelism for REORG TABLESPACE SHRLEVEL NONE use RESTART(PHASE) to restart at the beginning of the UNLOAD and RELOAD phases.
- Jobs that reorganize LOB table spaces use RESTART(PHASE).

If you restart a REORG job of one or more of the catalog or directory table spaces in the preceding list, you cannot use RESTART(CURRENT).

If you restart REORG in the UTILINIT phase, it re-executes from the beginning of the phase. If REORG abnormally terminates or system failure occurs while it is in the UTILTERM phase, you must restart the job with RESTART(PHASE).

The following table provides information about restarting REORG TABLESPACE, depending on the phase that REORG was in when the job stopped. For each phase of REORG and for each type of REORG TABLESPACE (with SHRLEVEL NONE, with SHRLEVEL REFERENCE, and with SHRLEVEL CHANGE), the following table indicates the types of restarts that are allowed (CURRENT and PHASE). A value of None indicates that no restart is allowed. The “Data Sets Required” column lists the data sets that must exist to perform the specified type of restart in the specified phase.
Table 95. REORG TABLESPACE utility restart information for SHRLEVEL NONE, REFERENCE, and CHANGE

<table>
<thead>
<tr>
<th>Phase</th>
<th>Type of restart allowed for SHRLEVEL</th>
<th>Type of restart allowed for SHRLEVEL</th>
<th>Type of restart allowed for SHRLEVEL</th>
<th>Required data sets</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNLOAD</td>
<td>CURRENT, PHASE</td>
<td>CURRENT, PHASE</td>
<td>None</td>
<td>SYSREC</td>
<td></td>
</tr>
<tr>
<td>RELOAD</td>
<td>CURRENT, PHASE</td>
<td>CURRENT, PHASE</td>
<td>None</td>
<td>SYSREC</td>
<td>1, 2</td>
</tr>
<tr>
<td>SORT</td>
<td>CURRENT, PHASE</td>
<td>CURRENT, PHASE</td>
<td>None</td>
<td>None</td>
<td>2, 3</td>
</tr>
<tr>
<td>BUILD</td>
<td>CURRENT, PHASE</td>
<td>CURRENT, PHASE</td>
<td>None</td>
<td>None</td>
<td>2, 3, 4</td>
</tr>
<tr>
<td>SORTBLD</td>
<td>CURRENT, PHASE</td>
<td>CURRENT, PHASE</td>
<td>None</td>
<td>None</td>
<td>2</td>
</tr>
<tr>
<td>LOG</td>
<td>Phase does not occur</td>
<td>Phase does not occur</td>
<td>None</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>SWITCH</td>
<td>Phase does not occur</td>
<td>CURRENT, PHASE</td>
<td>CURRENT, PHASE</td>
<td>Originals and shadows</td>
<td>3, 5</td>
</tr>
</tbody>
</table>

Note:
1. For None, if you specify NOSYSREC, restart is not possible, and you must execute the RECOVER TABLESPACE utility for the table space or partition. For REFERENCE, if the REORG job includes both SORTDATA and NOSYSREC, RESTART or RESTART(PHASE) restarts at the beginning of the UNLOAD phase.
2. If you specify SHRLEVEL NONE or SHRLEVEL REFERENCE, and the job includes the SORTKEYS option, use RESTART or RESTART(PHASE) to restart at the beginning of the RELOAD phase.
3. You can restart the utility with RESTART or RESTART(PHASE). However, because this phase does not take checkpoints, RESTART restarts from the beginning of the phase.
4. If you specify the PART option with REORG TABLESPACE, you cannot restart the utility at the beginning of the BUILD phase if any nonpartitioning index is in a page set REBUILD-pending (PSRBD) status.
5. If you specify REORG TABLESPACE SHRLEVEL REFERENCE PART with one or more nonpartitioned indexes, restart is allowed only in the SWITCH phase.
6. For REORG TABLESPACE with SHRLEVEL REFERENCE and PART, if a nonpartitioned index is defined on the table space, REORG TABLESPACE cannot be restarted before the SWITCH phase.

If you restart a REORG STATISTICS job by using RESTART CURRENT, inline statistics are not collected. To update catalog statistics, run the RUNSTATS utility after the restarted job completes. Restarting a REORG STATISTICS job with RESTART(PHASE) is conditional after executing UNLOAD PAUSE. To determine if catalog table statistics are going to be updated, see the following table. This table shows whether or not statistics are updated for REORG STATISTICS jobs according to the phase in which the job terminated and the restart value that was used.

Table 96. Statistics collection for REORG TABLESPACE utility phase restart

<table>
<thead>
<tr>
<th>Phase</th>
<th>CURRENT</th>
<th>PHASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTILINIT</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>UNLOAD</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>RELOAD</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>SORT</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>BUILD</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>SORTBLD</td>
<td>NO</td>
<td>YES</td>
</tr>
</tbody>
</table>
Review of REORG TABLESPACE output

The output from the REORG TABLESPACE utility consists of a reorganized table space, partition, or a range of partitions.

The following table summarizes the effect of REORG on a table space partition and on the corresponding index partition.

Table 97. Summary of the results of REORG TABLESPACE according to the type of specification.

<table>
<thead>
<tr>
<th>Specification</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>REORG TABLESPACE</td>
<td>All data + entire partitioning index + all nonpartitioning indexes</td>
</tr>
<tr>
<td>REORG TABLESPACE</td>
<td>Data for PART n + PART n of the partitioning index + index entries</td>
</tr>
<tr>
<td>PART n</td>
<td>for PART n in all nonpartitioning indexes</td>
</tr>
<tr>
<td>REORG TABLESPACE</td>
<td>Data for PART n1 through n2 + PART n1 through n2 of the</td>
</tr>
<tr>
<td>PART n1:n2</td>
<td>partitioning index + index entries for those partitions in all</td>
</tr>
<tr>
<td></td>
<td>nonpartitioning indexes</td>
</tr>
<tr>
<td>REORG TABLESPACE</td>
<td>Specified table space or partitions that are in REORG-pending status.</td>
</tr>
<tr>
<td>SCOPE PENDING</td>
<td></td>
</tr>
<tr>
<td>REORG SHRLEVEL</td>
<td>Unloads and builds entire NPIs, so that the NPI is largely</td>
</tr>
<tr>
<td>CHANGE PART and</td>
<td>reorganized even though only a part of the data is actually</td>
</tr>
<tr>
<td>SHRLEVEL</td>
<td>reorganized.</td>
</tr>
<tr>
<td>REFERENCE PART</td>
<td></td>
</tr>
</tbody>
</table>

When reorganizing a segmented table space, REORG leaves free pages and free space on each page in accordance with the current values of the FREEPAGE and PCTFREE parameters. (You can set those values by using the CREATE TABLESPACE, ALTER TABLESPACE, CREATE INDEX, or ALTER INDEX statements). REORG leaves one free page after reaching the FREEPAGE limit for each table in the table space. When reorganizing a nonsegmented table space, REORG leaves one free page after reaching the FREEPAGE limit, regardless of whether the loaded records belong to the same or different tables.

After running REORG TABLESPACE

Certain activities might be required after you run the REORG TABLESPACE utility, depending on your situation.

After a reorganization is complete, perform the following actions:

- If you have used LOG YES, consider taking an image copy of the reorganized table space or partition to:
  - Provide a full image copy for recovery. This action prevents the need to process the log records that are written during reorganization.
  - Permit making incremental image copies later.

You might not need to take an image copy of a table space for which all the following statements are true:

- The table space is relatively small.
- The table space is used only in read-only applications.
The table space can be easily loaded again in the event of failure.

In addition, you do not need to take an image copy if you used COPYDDN or FCCOPYDDN to take an inline image copy when you ran REORG.

- Use the RUNSTATS utility on the table space and its indexes if inline statistics were not collected, so that the Db2 catalog statistics take into account the newly reorganized data, and SQL paths can be selected with accurate information. You need to run RUNSTATS on nonpartitioning indexes only if you reorganized a subset of the partitions.
- If you use REORG TABLESPACE SHRLEVEL CHANGE with a mapping table, you can drop the mapping table and its index.
- If you use SHRLEVEL REFERENCE or CHANGE, and a table space, partition, or index resides in user-managed data sets, you can delete the user-managed shadow data sets.
- If you specify DISCARD on a REORG of a table that is involved in a referential integrity set, you need to run CHECK DATA for any affected referentially related objects that were placed in CHECK-pending status.

**Related reference:**
[Chapter 14, “COPY,” on page 133](#)

---

**Effects of running REORG TABLESPACE**

Running the REORG TABLESPACE utility can have effects on version numbers and the version of the data, control intervals, row formats, and table spaces that are defined with the NOT LOGGED attribute.

**The effect of REORG TABLESPACE on table space versions**

Db2 stores the range of used version numbers in the OLDEST_VERSION and CURRENT_VERSION columns of one or more of the following catalog tables, depending on the object:

- `SYSIBM.SYSTABLESPACE`
- `SYSIBM.SYSTABLESPACEPART`
- `SYSIBM.SYSINDEXES`
- `SYSIBM.SYSINDEXEXPART`

The OLDEST_VERSION column contains the oldest used version number, and the CURRENT_VERSION column contains the current version number.

When you run REORG TABLESPACE, the utility sets all of the rows in the table or partition to the format of the current object version. The utility also updates the range of used version numbers for indexes that are defined with the COPY NO attribute. REORG TABLESPACE sets the OLDEST_VERSION column equal to the CURRENT_VERSION column in the appropriate catalog row. These updated values indicate that only one version is active. Db2 can then reuse all of the other version numbers.

Recycling of version numbers is required when all of the version numbers are being used. All version numbers are being used when one of the following situations is true:

- The value in the CURRENT_VERSION column is one less than the value in the OLDEST_VERSION column.
- The value in the CURRENT_VERSION column is 255 for table spaces or 15 for indexes, and the value in the OLDEST_VERSION column is 0 or 1.
When REORG TABLESPACE runs, if a system page is missing for a table that is in version 0 format, the utility adds a system page for version 0 of the table. When REORG TABLESPACE runs on a partitioned table space that is at version 0, the utility adds missing version 0 system pages to each partition. When REORG TABLESPACE runs on a subset of partitions, and there is not enough space to insert a system page, the utility fails.

You can also run LOAD REPLACE, REBUILD INDEX, or REORG INDEX to recycle version numbers for indexes that are defined with the COPY NO attribute. To recycle version numbers for indexes that are defined with the COPY YES attribute or for table spaces, run MODIFY RECOVERY.

**The effect of REORG TABLESPACE on the control interval**

When you run REORG TABLESPACE without the REUSE option and the target data set is managed by Db2, Db2 deletes this data set before REORG processing begins. Db2 then redefines a new data set with a control interval that matches the page size.

**The effect of REORG TABLESPACE on row format**

When you run REORG with the ROWFORMAT RRF option on a table space or partition that is in basic row format, REORG converts that table space or partition to reordered row format. If the ROWFORMAT BRF option is specified, existing basic row format table spaces are not converted to reordered row format. If there is a table in the table space with an EDITPROC or VALIDPROC, the table space or partition remains in basic row format after the REORG.

If you run REORG on a catalog or directory table space, the catalog or directory table space remains in basic row format.

You can run REORG TABLESPACE on table spaces that contain some partitions in basic row format and some partitions in reordered row format. In this case, the utility converts the partitions that are in basic row format to reordered row format.

REORG TABLESPACE converts a undefined table space to reordered row format if both of the following conditions are true:

- ROWFORMAT RRF is explicitly specified.
- The specified target is an entire undefined table space in basic row format.

In this case, Db2 updates the row format definition in the catalog and directory. No data sets are defined for the table space.

**The effect of REORG on table spaces that are defined with NOT LOGGED attribute**

The following table shows the effect of REORG on table spaces that are defined with the NOT LOGGED attribute.
Table 98. REORG parameters

<table>
<thead>
<tr>
<th>LOAD REORG LOG keyword</th>
<th>Table space logging attribute</th>
<th>Table space type</th>
<th>What is logged after utility completes</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOG YES</td>
<td>NOT LOGGED</td>
<td>Non-LOB</td>
<td>LOG YES changes to LOG NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>No pending status or ICPY-pending1</td>
</tr>
<tr>
<td>LOG YES</td>
<td>NOT LOGGED</td>
<td>LOB</td>
<td>control information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>No pending status</td>
</tr>
<tr>
<td>LOG NO</td>
<td>NOT LOGGED</td>
<td>Non-LOB</td>
<td>nothing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>No pending status or ICPY-pending1</td>
</tr>
<tr>
<td>LOG NO</td>
<td>NOT LOGGED</td>
<td>LOB</td>
<td>nothing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>No pending status</td>
</tr>
</tbody>
</table>

Note:
1. The table space is set to ICPY-pending status if the records are discarded and no pending status is the records are not discarded.

Related tasks:

- Removing in-use table space versions (Db2 Administration Guide)

Sample REORG TABLESPACE control statements

Use the sample control statements as models for developing your own REORG TABLESPACE control statements.

**Example 1: Reorganizing a table space.**

The following control statement specifies that the REORG TABLESPACE utility is to reorganize table space DSN8S11D in database DSN8D11A.

```plaintext
REORG TABLESPACE DSN8D11A.DSN8S11D
```

**Example 2: Reorganizing a table space and specifying the unload data set**

The following control statement specifies that REORG TABLESPACE is to reorganize table space DSN8D81A.DSN8S81D. The DD name for the unload data set is UNLD, as specified by the UNLDDN option.

```plaintext
//STEP1 EXEC DSNUPROC,UID='IUJLU101.REORG',
// UTPROC='',
// SYSTEM='DSN'
//UTPRINT DD SYSOUT=*
//UNLD DD DSN=IUJLU101.REORG.STEP1.UNLD,DISP=(MOD,DELETE,CATLG),
// UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SORTWK01 DD DSN=IUJLU101.REORG.STEP1.SORTWK01,DISP=(MOD,DELETE,CATLG),
// UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SORTWK02 DD DSN=IUJLU101.REORG.STEP1.SORTWK02,DISP=(MOD,DELETE,CATLG),
// UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSIN DD *
//REORG TABLESPACE (DSN8D11A.DSN8S11D)
//UNLD (UNLD)
//*
```

Figure 62. Example REORG TABLESPACE control statement with the UNLDDN option
Example 3: Reorganizing a table space partition

The following control statement specifies that REORG TABLESPACE is to reorganize partition 3 of table space DSN8D11A.DSN8S11E. The SORTDEVT option indicates the device type for the temporary data sets that are to be dynamically allocated by the sort program.

```
REORG TABLESPACE DSN8D11A.DSN8S11E
  PART 3
  SORTDEVT SYSDA
```

Example 4: Reorganizing a table and using parallel index build

The following control statement specifies that REORG TABLESPACE is to reorganize table space DSNDB04.DSN8S81D and to use a parallel index build to rebuild the indexes. The indexes are built in parallel, because more than one index needs to be built and the job allocates the data sets that the sort program needs. Note that you no longer need to specify SORTKEYS; it is the default.

The job allocates the sort work data sets in two groups, which limits the number of pairs of utility subtasks to two. This example does not require UTPRINT DD statements because it uses DSNUPROC to invoke utility processing. DSNUPROC includes a DD statement that allocates UTPRINT to SYSOUT.

LOG NO specifies that records are not to be logged during the RELOAD phase. This option puts the table space in COPY-pending status.

```
/*
Chapter 28. REORG TABLESPACE
*/

//SAMPJOB  JOB  ..., //STEP1  EXEC  DSNUPROC,UID='SAMPJOB.REORG',UTPROC='',SYSTEM='DSN'
//SYSREC  DD  DSN=SAMPJOB.REORG.STEP1.SYSREC,DISP=(NEW,DELETE,CATLG),
  //UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
  /* First group of sort work data sets for parallel index build
//SW01WK01  DD  UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//SW01WK02  DD  UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//SW01WK03  DD  UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
  /* Second group of sort work data sets for parallel index build
//SW02WK01  DD  UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//SW02WK02  DD  UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//SW02WK03  DD  UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
  /* Sort work data sets for use by SORTDATA
//SORTWK01  DD  UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//SORTWK02  DD  UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//SORTWK03  DD  UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//SYSIN  DD  *
REORG TABLESPACE DSNDB04.DSN8S11D LOG NO
/*

Figure 63. Example REORG TABLESPACE control statement with LOG NO option

Example 5: Reorganizing a table while allowing read-write access

The following control statement specifies that REORG TABLESPACE is to reorganize table space DSNDB04.DSN8S81E and to use a parallel index build to rebuild the indexes. The sort program dynamically allocates sort work data sets. This example does not require UTPRINDD DD statements because it uses DSNUPROC to invoke utility processing. DSNUPROC includes a DD statement that allocates UTPRINT to SYSOUT. The SORTDEVT option indicates the device type for the temporary data sets that are to be dynamically allocated by the sort program. The SHRLEVEL CHANGE option specifies that while the table is being reorganized, users have read-write access. The name of the mapping table is
DSN8MAP. This table is used to map the RIDs of data records in the original copy of the area to the corresponding RIDs in the shadow copy.

```
//SAMPJOB  JOB ...
//STEP1  EXEC DSNUPROC,UID='SAMPJOB.REORG',UTPROC='',SYSTEM='DSN'
//SYSCOPY  DD UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND),
//  DSN=SAMPJOB,COPY,DISP=(NEW,CATLG,CATLG)
//SYSSIN  DD *
REORG TABLESPACE DSN8D81A.DSN8S11D LOG NO SORTDEVT SYSDA SORTNUM 4
  SHRLEVEL CHANGE MAPPINGTABLE DSN8MAP
/*
```

### Example 6: Specifying a deadline for the SWITCH phase while reorganizing a table

The following control statement specifies that REORG TABLESPACE is to reorganize table space DSN8D81A.DSN8S11D. The DEADLINE option indicates that the deadline for start of the SWITCH phase is eight hours from the start of the REORG job. The COPYDDN and RECOVERYDDN options indicate that the utility is to take an image copy of the table space. Db2 is to write the primary image copy at the local site to a data set that is defined by the MYCOPY1 DD statement and to write the primary image copy at the recovery site to a data set that is defined by the MYCOPY2 DD statement. SHRLEVEL REFERENCE indicates that access is restricted during reorganization.

```
REORG TABLESPACE DSN8D81A.DSN8S11D COPYDDN(MYCOPY1)
  RECOVERYDDN(MYCOPY2) SHRLEVEL REFERENCE
  DEADLINE CURRENT TIMESTAMP + 8 HOURS
```

### Example 7: Setting a deadline for a REORG TABLESPACE job

The following control statement specifies that REORG TABLESPACE is to reorganize table space DSN8D11A.DSN8S11D. The DEADLINE option indicates that the deadline for the start of the SWITCH phase is eight hours from the start of the REORG job. The name of the mapping table is DSN8810.MAP_TBL. The maximum amount of time for log processing in the read-only (last) iteration of log processing is 240 seconds, as indicated by the MAXRO option. If Db2 is not reading the log quickly enough after the applications write to the log, Db2 drains the write claim class after sending the LONGLOG message to the operator. That draining takes place at least 900 seconds after the LONGLOG message is sent, as indicated by the DELAY option. Db2 is also to take inline image copies for the local site and recovery site, as indicated by the COPYDDN and RECOVERYDDN options.

```
REORG TABLESPACE DSN8D11A.DSN8S11D COPYDDN(MYCOPY1)
  RECOVERYDDN(MYCOPY2) SHRLEVEL REFERENCE
  DEADLINE CURRENT TIMESTAMP + 8 HOURS
  MAPPINGTABLE DSN8810.MAP_TBL MAXRO 240 LONGLOG DRAIN DELAY 900
```

### Example 8: Reorganizing a range of table space partitions

The following control statement specifies that REORG TABLESPACE is to reorganize partitions 3 through 5 of table space DSN8D11A.DSN8S11E. The SORTDEVT option indicates the device type for the temporary data sets that are to be dynamically allocated by the sort program. The SHRLEVEL NONE option indicates that while the data is being unloaded, applications can read but can't write. While the data is being reloaded, applications can have read-write access. SHRLEVEL NONE is the default. The COPYDDN option indicates that the utility is to take an image copy of the table space and to write the primary image copy to the data set that is defined by the SYSCOPY DD statement.
REORG TABLESPACE DSN8D11A.DSN8S11E
PART 3:5
  SORTDEVTSYSDA
  SHLEVEL NONE
  COPYDDN SYSCOPY

Example 9: Reorganizing a partition and updating the statistics

The following control statement specifies that REORG TABLESPACE is to reorganize partition 3 of table space DSN8D11A.DSN8S11E. The STATISTICS option indicates that the utility is also to update statistics in the catalog for that partition. Note that the STATISTICS option is not valid for LOB table spaces.

REORG TABLESPACE DSN8D11A.DSN8S11E
  STATISTICS PART 3

Example 10: Reorganizing a table space and reporting table space and index statistics

The following control statement specifies that REORG TABLESPACE is to reorganize table space DSN8D11A.DSN8S11E. The SORTDATA option indicates that the data is to be unloaded and sorted in clustering order. This option is the default and does not need to be specified. The STATISTICS, TABLE, INDEX, and REPORT YES options indicate that the utility is also to report catalog statistics for all tables in the table space and for all indexes that are defined on those tables. The FREQVAL, NUMCOLS, and COUNT options indicate that Db2 is to collect 10 frequent values on the first key column of the index. UPDATE NONE indicates that the catalog tables are not to be updated. This option requires that REPORT YES also be specified. Because both STATISTICS and INDEX are specified, the utility also collects statistics on the values in the key columns of indexes.

REORG TABLESPACE DSN8D11A.DSN8S11E SORTDATA STATISTICS
  TABLE INDEX(ALL) FREQVAL NUMCOLS 1
  COUNT 10 REPORT YES UPDATE NONE

Example 11: Determining whether a table space should be reorganized

The following control statement specifies that REORG TABLESPACE is to report if the OFFPOSLIMIT and INDREFLIMIT values for partition 11 of table space DBHR5201.TPHR5201 exceed the specified values (11 for OFFPOSLIMIT and 15 for INDREFLIMIT).
On successful completion, Db2 returns output that is similar to the output in the following sample output. This sample output shows that the limits have been met.

**Example 12: Conditionally reorganizing a table space**

In the following example, the RUNSTATS utility control statement specifies that the utility is to update space statistics in the catalog for table space DBHR5201.TPHR5201. This RUNSTATS job ensures that the space statistics for this table space are current. The subsequent REORG TABLESPACE control statement specifies that if any of the values for OFFPOSLIMIT or INDREFLIMIT exceed 9, the utility is to reorganize the table space.
On successful completion, Db2 returns output for the REORG TABLESPACE job that is similar to the output in the following sample output.

```sql
DSNU050I DSNUUGUTC - REORG TABLESPACE DBHR5201.TPHR5201 SHRLEVEL CHANGE MAPPINGTABLE
MAP1 COPYDDN(SYSINDEX1)
OFFPOSLIMIT 9 INDEFRELF 9
DSNU286I = DSNULIMIT - REORG TABLESPACE DBHR5201.TPHR5201 OFFPOSLIMIT SYSINDEXPART ROWS
* CREATOR.IXNAME : ADMF001.IPHR5201
CREATOR.TBNAME : ADMF001.TPHR5201
    PART: 1 CARD: 3.6E+01 FAROFFPOSF: 0.0E0 NEAROFFPOSF: 1.2E+01
STATTIME: 2002-05-28-16.22.18
CREATOR.IXNAME : ADMF001.IPHR5201
CREATOR.TBNAME : ADMF001.TPHR5201
    PART: 2 CARD: 5.0E+00 FAROFFPOSF: 0.0E0 NEAROFFPOSF: 0.0E0
STATTIME: 2002-05-28-16.22.18
...
* CREATOR.IXNAME : ADMF001.IPHR5201
CREATOR.TBNAME : ADMF001.TPHR5201
    PART: 11 CARD: 6.758E+03 FAROFFPOSF: 2.892E+03 NEAROFFPOSF: 8.18E+02
STATTIME: 2002-05-28-16.22.18
DSNU287I = DSNULIMIT - REORG TABLESPACE DBHR5201.TPHR5201 INDEFRELF SYSTABLEPART ROWS
```

---

Figure 66. Example of conditionally reorganizing a table

On successful completion, Db2 returns output for the REORG TABLESPACE job that is similar to the output in the following sample output.
Example 13: Reorganizing a table space after waiting for SQL statements to complete.

The following control statement specifies that REORG TABLESPACE is to reorganize the table space in the REORG_TBS list, which is defined in the preceding LISTDEF utility control statement. Before reorganizing the table space, REORG TABLESPACE is to wait for 30 seconds for SQL statements to finish adding or changing data. This interval is indicated by the DRAIN_WAIT option. If the SQL statements do not finish, the utility is to try again up to four times, as indicated by the RETRY option. The utility is to wait 10 seconds between retries, as indicated by the RETRY_DELAY option.

The TEMPLATE utility control statements define the data set characteristics for the data sets that are to be dynamically allocated during the REORG TABLESPACE job. The OPTIONS utility control statement indicates that the TEMPLATE statements and LISTDEF statement are to run in PREVIEW mode.
//STEP1  EXEC DSNUPROC,UID='HUHRU257.REORG',TIME=1440,
//       UTPROC='',
//       SYSTEM='DSN'
//UTPRINT DD SYSOUT=*  
//SYSIN DD *

OPTIONS PREVIEW
TEMPLATE CPYTMP UNIT(SYSDA)
  DSN(HUHRU257.REORG.TPHR5701.TPSYSCOPY1)
TEMPLATE SRC
  UNIT(SYSDA) DISP(NW,CATLG,CATLG)
  DSN(HUHRU257.REORG.&ST..SDISC)
TEMPLATE SDISC
  UNIT(SYSDA) DISP(NW,CATLG,CATLG)
  DSN(HUHRU257.REORG.&ST..SREC)
TEMPLATE SPUNCH
  UNIT(SYSDA) DISP(NW,CATLG,CATLG)
  DSN(HUHRU257.REORG.&ST..SREC)
LISTDEF REORG_TBSP INCLUDE TABLESPACE DBHR5701.TPHR5701
OPTIONS OFF
REORG TABLESPACE LIST REORG_TBSP
DRAIN WAIT 30  RETRY 4  RETRY_DELAY 10
STATISTICS
TABLE (ALL) SAMPLE 60
INDEX (ALL FREEQUAL NUMCOLS 2 COUNT 15)
SHRLEVEL CHANGE MAPPINGTABLE MAP5702
LONGLOG DRAIN MAXRO DEFER DELAY 30
COPYDDN (CPYTMP)
SORTDEV SYSDA SORTNUM 8
PUNCHDDN SPUNCH
DISCARD DD SYSDA
UNLDDN SREC

Figure 67. Example of reorganizing a table space by using DRAIN WAIT, RETRY, and RETRY_DELAY

On successful completion, Db2 returns output similar to the output in the following sample output.

Chapter 28. REORG TABLESPACE 661
<table>
<thead>
<tr>
<th>PART</th>
<th>NUMBER OF RECORDS</th>
<th>UNLOADED</th>
<th>SUCCESSFUL</th>
<th>FAILURE</th>
<th>ELAPSED TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>PART</td>
<td>RECORDS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2520</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>00:00:00</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**
- **PART 1** processed 2520 records, unloaded 0 records, failed 0 records, elapsed time 00:00:00.
- **PART 2** and onwards processed 0 records, unloaded 0 records.
- **SUCCESSFUL** indicates successful completion of each phase.
- **FAILURE** indicates any failures encountered during the processing.
- **ELAPSED TIME** shows the duration of each phase.
DSNUS1O1  0 280 14:57:37.45  DSNUUSPC - SYSCOLSTATS CATALOG UPDATE FOR ADMF001.IPFRS701 SUCCESSFUL
DSNUS1O1  0 280 14:57:37.45  DSNUUSPD - SYSCOLINDEXSTATS CATALOG UPDATE FOR ADMF001.IPFRS701 SUCCESSFUL
DSNUS1O1  0 280 14:57:37.46  DSNUUSIP - SYSCOLINDEXPART CATALOG UPDATE FOR ADMF001.IXR5702 SUCCESSFUL
DSNUS1O1  0 280 14:57:37.46  DSNUUSIF - SYSCOLINDEXPART CATALOG UPDATE FOR ADMF001.IXR5702 SUCCESSFUL
DSNUS1O1  0 280 14:57:37.47  DSNUUSIP - SYSCOLINDEXPART CATALOG UPDATE FOR ADMF001.IXR5704 SUCCESSFUL
DSNUS1O1  0 280 14:57:37.47  DSNUUSIF - SYSCOLINDEXPART CATALOG UPDATE FOR ADMF001.IXR5704 SUCCESSFUL
DSNUS1O1  0 280 14:57:37.48  DSNUUSI - SYSCOLINDEXES CATALOG UPDATE FOR ADMF001.IPFRS701 SUCCESSFUL
DSNUS1O1  0 280 14:57:37.48  DSNUUSIF - SYSCOLINDEXES CATALOG UPDATE FOR ADMF001.IPFRS701 SUCCESSFUL
DSNUS1O1  0 280 14:57:37.54  DSNUUSUC - SYSCOLUMNS CATALOG UPDATE FOR ADMF001.IXR5702 SUCCESSFUL
DSNUS1O1  0 280 14:57:37.54  DSNUUSUC - SYSCOLUMNS CATALOG UPDATE FOR ADMF001.IXR5702 SUCCESSFUL
DSNUS1O1  0 280 14:57:37.57  DSNUUSUC - SYSCOLUMNS CATALOG UPDATE FOR ADMF001.IXR5703 SUCCESSFUL
DSNUS1O1  0 280 14:57:37.57  DSNUUSUC - SYSCOLUMNS CATALOG UPDATE FOR ADMF001.IXR5703 SUCCESSFUL
DSNUS1O1  0 280 14:57:37.60  DSNUUSUC - SYSCOLUMNS CATALOG UPDATE FOR ADMF001.IXR5704 SUCCESSFUL
DSNUS1O1  0 280 14:57:37.60  DSNUUSUC - SYSCOLUMNS CATALOG UPDATE FOR ADMF001.IXR5704 SUCCESSFUL
DSNUS1O1  0 280 14:57:37.66  DSNUUSUC - SYSCOLUMNS CATALOG UPDATE FOR ADMF001.IXR5705 SUCCESSFUL
DSNUS1O1  0 280 14:57:37.66  DSNUUSUC - SYSCOLUMNS CATALOG UPDATE FOR ADMF001.IXR5705 SUCCESSFUL
DSNUS1O1  0 280 14:57:37.71  DSNUUSUC - SYSCOLUMNS CATALOG UPDATE FOR ADMF001.IXR5706 SUCCESSFUL
DSNUS1O1  0 280 14:57:37.71  DSNUUSUC - SYSCOLUMNS CATALOG UPDATE FOR ADMF001.IXR5706 SUCCESSFUL
DSNUS1O1  0 280 14:57:37.72  DSNUUSUC - SYSCOLUMNS CATALOG UPDATE FOR ADMF001.IXR5705 SUCCESSFUL
DSNUS1O1  0 280 14:57:37.72  DSNUUSUC - SYSCOLUMNS CATALOG UPDATE FOR ADMF001.IXR5705 SUCCESSFUL
DSNUS1O1  0 280 14:57:37.73  DSNUUSUC - SYSCOLUMNS CATALOG UPDATE FOR ADMF001.IXR5705 SUCCESSFUL
DSNUS1O1  0 280 14:57:37.73  DSNUUSUC - SYSCOLUMNS CATALOG UPDATE FOR ADMF001.IXR5705 SUCCESSFUL
DSNUS1O1  0 280 14:57:37.74  DSNUUSUC - SYSCOLUMNS CATALOG UPDATE FOR ADMF001.IXR5706 SUCCESSFUL
DSNUS1O1  0 280 14:57:37.74  DSNUUSUC - SYSCOLUMNS CATALOG UPDATE FOR ADMF001.IXR5706 SUCCESSFUL
DSNUS1O1  0 280 14:57:42.23  DSNUBUGAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE = 0

DSNU3941 = DSNUBAX - SORTBLD PHASE STATISTICS - NUMBER OF KEYS = 331 FOR INDEX ADMF001.IXR5706
DSNU3941 = DSNUBAX - SORTBLD PHASE STATISTICS - NUMBER OF KEYS = 331 FOR INDEX ADMF001.IXR5705
DSNU3941 = DSNUBAX - SORTBLD PHASE STATISTICS - NUMBER OF KEYS = 331 FOR INDEX ADMF001.IXR5705
DSNU3941 = DSNUBAX - SORTBLD PHASE STATISTICS - NUMBER OF KEYS = 331 FOR INDEX ADMF001.IXR5705
DSNU3941 = DSNUBAX - SORTBLD PHASE STATISTICS - NUMBER OF KEYS = 331 FOR INDEX ADMF001.IXR5705
DSNU3941 = DSNUBAX - SORTBLD PHASE STATISTICS - NUMBER OF KEYS = 331 FOR INDEX ADMF001.IXR5705
DSNU3941 = DSNUBAX - SORTBLD PHASE STATISTICS - NUMBER OF KEYS = 331 FOR INDEX ADMF001.IXR5705
DSNU3941 = DSNUBAX - SORTBLD PHASE STATISTICS - NUMBER OF KEYS = 331 FOR INDEX ADMF001.IXR5705
DSNU3941 = DSNUBAX - SORTBLD PHASE STATISTICS - NUMBER OF KEYS = 331 FOR INDEX ADMF001.IXR5705

Chapter 28. REORG TABLESPACE  663
Example 14: Using a mapping table

In the following example, a mapping table and mapping table index are created. Then, a REORG TABLESPACE job uses the mapping table, and finally the mapping table is dropped. Some parts of this job use the EXEC SQL utility to execute dynamic SQL statements.

The first EXEC SQL control statement contains the SQL statements that create a mapping table that is named MYMAPPING_TABLE. The second EXEC SQL control statement contains the SQL statements that create mapping index MYMAPPING_INDEX on the table MYMAPPING_TABLE.

The REORG TABLESPACE control statement then specifies that the REORG TABLESPACE utility is to reorganize table space DSN8D81P.DSN8S81C and to use mapping table MYMAPPING_TABLE.

Finally, the third EXEC SQL statement contains the SQL statements that drop MYMAPPING_TABLE.

```
EXEC SQL
CREATE TABLE MYMAPPING_TABLE
  (TYPE CHAR( 01 ) NOT NULL,
   SOURCE_RID CHAR( 05 ) NOT NULL,
   TARGET_XRID CHAR( 09 ) NOT NULL,
   LRSN CHAR( 10 ) NOT NULL)
IN DSN8D81P.DSN8S81Q
CCSID EBCDIC
ENDEXEC
EXEC SQL
CREATE UNIQUE INDEX MYMAPPING_INDEX
ON MYMAPPING_TABLE
  (SOURCE_RID ASC,
   TYPE,
   TARGET_XRID,
   LRSN)
USING STOGROUP DSN8G710
PRIQTY 120 SECQTY 20
ERASE NO
BUFFERPOOL BP0
CLOSE NO
ENDEXEC
REORG TABLESPACE DSN8D81P.DSN8S81C
COPYDDN(COPYDDN)
SHRLEVEL CHANGE
DEADLINE CURRENT_TIMESTAMP+8 HOURS
MAPPINGTABLE MYMAPPING_TABLE
MAXRO 240 LONGLOG DRAIN DELAY 900
SORTDEVT SYSDA SORTNUM 4
```
Example 15: Discarding records from one table while reorganizing a table space

The following control statement specifies that REORG TABLESPACE is to reorganize table space DSN8D51A.DSN8S51E. During reorganization, records in table DSN8510.EMP are discarded if they have the value D11 in the WORKDEPT field. This discard criteria is specified in the WHEN clause that follows the DISCARD option. Because a SYSDISC DD statement is included in the JCL, any discarded rows are to be written to the data set that is identified by this DD statement.

The COPYDDN option specifies that during the REORG, Db2 is also to take an inline copy of the table space. This image copy is to be written to the data set that is identified by the SYSCOPY DD statement.

```
//REORGDIS EXEC DSNUPROC,TIME=1440,
 //         UTPROC='',
 //         SYSTEM='DSN',UID='REORGDIS.EMP'
//SYSREC DD DISP=(NEW,CATLG,CATLG),
 //        DSN=SYSADM.REORGDIS.SYSREC,
 //        UNIT=SYSDA,SPACE=(TRK,(15,15))
//SYSDISC DD DISP=(NEW,CATLG,CATLG),
 //        DSN=SYSADM.REORGDIS.SYSDISC,
 //        UNIT=SYSDA,SPACE=(TRK,(15,15))
//SYSPUNCH DD DISP=(NEW,CATLG,CATLG),
 //        DSN=SYSADM.REORGDIS.SYSPUNCH,
 //        UNIT=SYSDA,SPACE=(TRK,(15,15))
//SYSCOPY DD DISP=(NEW,CATLG,CATLG),
 //        DSN=SYSADM.DSN8051A.DSN8S51E.COPY,
 //        UNIT=SYSDA,SPACE=(TRK,(30,30)),
//SYSIN DD *
REORG TABLESPACE
DSN8081A.DSN8S81E
DISCARD
FROM TABLE DSN8810.EMP
WHEN (WORKDEPT = 'D11')
SHRLEVEL NONE COPYDDN SYSCOPY
```

Figure 68. Example REORG statement that specifies discard criteria

Example 16: Discarding records from multiple tables while reorganizing a table space

The following control statement specifies that REORG TABLESPACE is to reorganize table space DBKC0501.TLKC0501. During reorganization, the following records are discarded:

- Records in table TBKC0501 that have a value in the QT_INV_TRANSACTION column that is less than or equal to 700, and a value in the NO_DEPT column that is equal to X'33303230'.
- Records in table TBKC0502 that have a value in the NO_WORK_CENTER column that is equal to either X'333031303120' or X'333032303620'.
This discard criteria is specified with the DISCARD option. Any discarded rows are to be written to the SYSDISC data set, as specified by the DISCARDDN option.

//STEP1 EXEC DSNUPROC,UID='IUKCU105.REORG2',
     UTPROC='',
     SYSTEM='SSTR'
//UPRINT DD SYSOUT=* 
//SYSDISC DD DSN=IUKCU105.REORG2.STEP1.SYSDISC,
     DISP=(MOD,DELETE,CATLG),
     UNIT=SYSDA,SPACE=(2000,(20,20),,,ROUND),
     DCB=(RECFM=FB,LRECL=80,BLKSIZE=2400)
//SYSSREC DD DSN=IUKCU105.REORG2.STEP1.SYSSREC,
     DISP=(MOD,DELETE,CATLG),
     UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSCOPY DD DSN=IUKCU105.REORG2.STEP1.SYSCOPY,
     DISP=(MOD,DELETE,CATLG),
     UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//LOADSTMT DD DSN=IUKCU105.REORG2.STEP1.SYSPUNCH,
     DISP=(MOD,DELETE,CATLG),
     UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSIN DD *
    REORG TABLESPACE DBKQAA01.TPKQAA01 SHLEVEL REFERENCE
    PUNCHDDN LOADSTMT DISCARDDN SYSDISC
    UNLOAD CONTINUE
    DISCARD
    FROM TABLE TBKC0501
    WHEN (QT_INV_TRANSACTION <= 700 AND
          NO_DEPT = 'X33303230')
    FROM TABLE TBKC0502
    WHEN (NO_WORK_CENTER = 'X333031303120' OR
          NO_WORK_CENTER = 'X333032303620')
/*/ 

Figure 69. Example REORG statement that specifies discard criteria for several tables

**Example 17: Reorganizing only those partitions that are in REORG-pending status**

The following control statement specifies that REORG TABLESPACE is to reorganize only those partitions of table space DBKQAA01.TPKQAA01 that are in the range from 2 to 10 and are in REORG-pending status.

//STEP1 EXEC DSNUPROC,UID='JUKQU1AA.REORG6',
     UTPROC='',SYSTEM='SSTR'
//SYSSREC DD DSN=JUKQU1AA.REORG6.STEP1.SYSSREC,
     DISP=(MOD,DELETE,CATLG),
     UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSCOPY DD DSN=JUKQU1AA.REORG6.STEP1.SYSCOPY,
     DISP=(MOD,DELETE,CATLG),
     UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSSUT1 DD DSN=JUKQU1AA.REORG6.STEP1.SYSSUT1,
     DISP=(MOD,DELETE,CATLG),
     UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SORTOUT DD DSN=JUKQU1AA.REORG6.STEP1.SORTOUT,
     DISP=(MOD,DELETE,CATLG),UNIT=SYSDA,
     SPACE=(4000,(20,20),,,ROUND)
//SYSIN DD *
    REORG TABLESPACE DBKQAA01.TPKQAA01 SCOPE PENDING PART 2:10
/*/ 

Figure 70. Example REORG TABLESPACE statement with SCOPE PENDING
Example 18: Creating a FlashCopy image copy with REORG TABLESPACE

The following REORG TABLESPACE utility control statement reorganizes table space DSN8SA1D and creates a FlashCopy image copy.

```
//SYSADMA JOB (ACCOUNT), 'NAME', NOTIFY=&SYSUID
//*
//UTIL EXEC DSNUPROC,SYSTEM=VA1A,UID='TEMP',UTPROC=''
//DSNUPROC.SYSREC DD DSN=SYSOPS.DSNAME,
// DISP=(NEW,DELETE),
// SPACE=(CYL,(20,20),RLSE),
// UNIT=SYSDA,VOL=SER=SCR03
//DSNUPROC.SYSUT1 DD DSN=SYSOPS.SYSUT1,
// DISP=(NEW,DELETE),
// SPACE=(CYL,(9,90),RLSE),
// UNIT=SYSDA,VOL=SER=SCR03
//DSNUPROC.SYSIN DD *
LISTDEF COPY_LIST INCLUDE TABLESPACE DSN8SA1A.DSN8SA1D
TEMPLATE SCOPY UNIT(SYSDA) DISP(NEW,CATLG,DELETE)
DSN(DSNT1.&DB..&TS..CPY1.D&TIME.)
TEMPLATE FCOPY UNIT(SYSDA) DISP(NEW,CATLG,DELETE)
DSN(DSNFC.&DB..&TS..P&PA..D&TIME.)
REORG TABLESPACE LIST COPY_LIST SHRLEVEL REFERENCE FLASHCOPY YES
FCCOPYDDN(FCOPY) COPYDDN(SCOPY)
```

Example 19: Reorganizing only clone tables

The REORG TABLESPACE control statement indicates that REORG TABLESPACE is to reorganize only clone tables from the specified table spaces.

```
REORG TABLESPACE DBKQBS01.TPKQBS01 CLONE
```

Related reference:
- CREATE INDEX (Db2 SQL)
- CREATE TABLE (Db2 SQL)
- DROP (Db2 SQL)
- Db2 Sort for z/OS

Related information:
- DFSORT Application Programming Guide
Chapter 29. REPAIR

The REPAIR online utility repairs data. The data can be your own data or data that you would not normally access, such as space map pages and index entries.

You use REPAIR to replace invalid data with valid data. Be extremely careful when using REPAIR. Improper use can damage the data even further.

You can use the REPAIR utility to:
- Test database definitions (DBDs)
- Repair DBDs
- Reset a pending status on a table space or index
- Verify the contents of data areas in table spaces and indexes
- Replace the contents of data areas in table spaces and indexes
- Delete a single row from a table space
- Produce a hexadecimal dump of an area in a table space or index
- Delete an entire LOB from a LOB table space
- Dump LOB pages
- Rebuild object descriptors (OBDs) for a LOB table space
- Manage version numbers
- Turn on or off Persistent Read Only (PRO) restricted status for table space partitions
- Check for and fix any inconsistencies between the information in the catalog and the data.
- Update the CURRENT_VERSION or OLDEST_VERSION column of the SYSIBM.SYSTABLESPACE table when the data in the table space is at a different versions from the versions that are recorded in the catalog.
- Insert missing system pages for table spaces or tables that are in version 0 format.
- Update the version numbers in the VERSION column of the SYSIBM.SYSTABLES catalog table for all tables in the table space that are at version 0 to the same version number as in the CURRENT_VERSION column in SYSIBM.SYSTABLESPACE.

Output

The output from the REPAIR utility can consist of one or more modified pages in the specified Db2 table space or index and a dump of the contents.

Authorization required

To execute this utility, you must use a privilege set that includes one of the following authorities:
- REPAIR privilege for the database
- DBADM or DBCTRL authority for the database. If the object on which the utility operates is in an implicitly created database, DBADM authority on the implicitly created database or DSNDB04 is required.
- DATAACCESS authority
• SYSCTRL or SYSADM authority

An ID with installation SYSOPR authority can also execute REPAIR, but only on a table space in the DSNDB01 or DSNDB06 database.

To execute REPAIR with the DBD option, you must use a privilege set that includes SYSADM, SYSCTRL, or installation SYSOPR authority.

REPAIR should be used only by a person that is knowledgeable in Db2 and your data. Grant REPAIR authorization only to the appropriate people.

**Execution phases of REPAIR**

The phases for REPAIR are:

<table>
<thead>
<tr>
<th>Phase</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTILINIT</td>
<td>Performs initialization</td>
</tr>
<tr>
<td>REPAIR</td>
<td>Repairs data</td>
</tr>
<tr>
<td>UTILTERM</td>
<td>Performs cleanup</td>
</tr>
</tbody>
</table>

**Syntax and options of the REPAIR control statement**

The REPAIR utility control statement, with its multiple options, defines the function that the utility job performs.

You can create a control statement with the ISPF/PDF edit function. After creating it, save it in a sequential or partitioned data set. When you create the JCL for running the job, use the SYSIN DD statement to specify the name of the data set that contains the utility control statement.

**REPAIR syntax diagram**

```
REPAIR
  OBJECT
  LOG YES
  LOG NO
  locate block
    set statement
    dbd-statement
    level-id statement
    versions statement
    catalog statement
    system-pages statement
  CLONE
```

locate block:
locate-table-space-spec:

```
LOCATE
locate-table-space-spec
LOCATE
locate-index-spec
LOCATE
LIB-table-space-spec
LOCATE
xml-table-space-spec
```

```
locate-table-space-spec:

TABLESPACE
table-space-name
PAGE-integer'
PART-integer
RID-integer'
KEY-literal
INDEX-index-name
```

```
locate-index-spec:

INDEX-index-name
INDEXSPACE-index-space-name
PAGE-integer'
PART-integer
```

```
LOB-table-space-spec:

TABLESPACE
database-name.
table-space-name
ROWID-
X-byte-string'
```

```
VERSION-X-byte-string'
delete statement
dump statement
```

xml-table-space-spec:

```
TABLESPACE
xml-table-space-name
DOCID-X-byte-string'
```

```
delete statement
```

verify statement:

```
Chapter 29. REPAIR  671
```
dbd statement:

```
DBD
  DROP DATABASE database-name DBID X'dbid'
  TEST
  DIAGNOSE DATABASE database-name
  REBUILD
  OUTDDN ddname
```

level-id statement:

```
LEVELID
  TABLESPACE database-name table-space-name
  INDEX index-name
  INDEXSPACE index-space-name
```

versions statement:

```
VERSIONS
  TABLESPACE database-name table-space-name
  INDEX index-name
  INDEXSPACE index-space-name
```

catalog statement:
system-pages statement:

REPAIR option descriptions

OBJECT
Indicates that an object is to be repaired. This keyword is optional.

LOG
Indicates whether the changes that REPAIR makes are to be logged. If the changes are to be logged, they are applied again if the data is recovered.

YES
Indicates that the changes are to be logged.

REPAIR LOG YES cannot override the LOG NO attribute of a table space.

NO
Indicates that the changes are not to be logged. You cannot use this option with a DELETE statement.

REPAIR LOG NO can override the LOG YES attribute of a table space.

TABLESPACE database-name.table-space-name
Specifies the table space (and, optionally, the database to which it belongs) whose level identifier is to be reset (if you specify LEVELID) or whose version identifier is to be updated (if you specify VERSIONS).

database-name
Specifies the name of the database to which the table space belongs.

The default value is DSND04.

table-space-name
Specifies the name of the table space.

INDEX
Specifies the index whose level identifier is to be reset (if you specify LEVELID) or whose version identifier is to be updated (if you specify VERSIONS).

creator-id
Specifies the creator of the index. Specifying this qualifier is optional.

index-name
Specifies the name of the index. Enclose the index name in quotation marks if the name contains a blank.

You can specify either INDEX or INDEXSPACE to identify an index. To specify multiple indexes, repeat the keyword.
INDEXSPACE
   Specifies the index space for the index whose level identifier is to be reset (if you specify LEVELID) or whose version identifier is to be updated (if you specify VERSIONS). You can obtain the index space name for an index from the SYSIBM.SYSINDEXES catalog table. The index space name must be qualified.

   database-name
      Specifies the name of the database to which the index space belongs.

   index-space-name
      Specifies the name of the index space.

   You can specify either INDEX or INDEXSPACE to identify an index. To specify multiple indexes, repeat the keyword.

PART
   Identifies a partition of the table space or index (including a partition of a data-partitioned secondary index).

   integer
      is the physical partition number. It must be in the range from 1 to the number of partitions that are defined for the table space. The maximum is 4096.

CLONE
   Indicates that REPAIR is to process only the specified objects that are table spaces that contain clone tables, indexes on clone tables, or index spaces that contain indexes on clone tables. If you specify CLONE, you cannot specify VERSIONS. Clones cannot be created for tables with active versions.

   If you specify SET with CLONE, the status is changed for only the specified table spaces and their indexes. The CLONE keyword applies to all SET statements and LOCATE statements within the same REPAIR utility control statement.

LOCATE block

   A LOCATE block is a set of statements, each with its own options, that begins with a LOCATE statement and ends with the next LOCATE or SET statement, or with the end of the job. You can include more than one LOCATE block in a REPAIR utility statement.

   In any LOCATE block, you can use VERIFY, REPLACE, or DUMP as often as you like; you can use DELETE only once.

LOCATE block: LOCATE TABLESPACE statement option descriptions

   The LOCATE TABLESPACE statement locates data that is to be repaired within a table space.

   One LOCATE statement is required for each unit of data that is to be repaired. Several LOCATE statements can appear after each REPAIR statement.

   If a REPAIR statement is followed by more than one LOCATE statement, all processing that is caused by VERIFY, REPLACE, and DUMP statements is committed before the next LOCATE statement is processed.
TABLESPACE
Specifies the base table space or XML table space (and, optionally, the database to which it belongs) in which data is to be located for repair.

database-name
The name of the database to which the base table space or XML table space belongs. This is optional.

table-space-name
The name of the base table space that contains the data to be repaired.

xml-table-space-name
The name of the XML table space that contains the data to be repaired.

PAGE
Specifies the relative or absolute page number within the table space, partitioned table space, or index that is to be operated on. The first page, in either case, is 0 (zero). For a table space with absolute page numbering, you can specify a PAGE keyword with a relative or absolute page number. For a table space with relative page numbering, you must specify a PAGE keyword with a relative page number.

integer
integer is a decimal number from one to six digits in length.

X'byte-string'
Specifies that the data of interest is an entire page. The specified offsets in byte-string and in subsequent statements are relative to the beginning of the page. The first byte of the page is at offset 0.

byte-string is a hexadecimal value from one to eight characters in length. You do not need to enter leading zeros. Enclose the byte-string between apostrophes, and precede it with X.

PART integer
Specifies the partition that contains the page that is to be located. Part is valid only for partitioned table spaces.

integer is the number of the partition.

RID X'byte-string'
Specifies that the data that is to be located is a single row. The specified offsets in byte-string and in subsequent statements are relative to the beginning of the row. The first byte of the stored row prefix is at offset 0.

byte-string can be a hexadecimal value from one to eight characters in length. You do not need to enter leading zeros. Enclose the byte string between apostrophes, and precede it with an X.

KEY literal
Specifies that the data that is to be located is a single row, identified by literal. The specified offsets in subsequent statements are relative to the beginning of the row. The first byte of the stored row prefix is at offset 0.

literal is any SQL constant that can be compared with the key values of the named index.

Character constants that are specified within the LOCATE KEY option cannot be specified as ASCII or Unicode character strings. No conversion of the values is performed. To use this option when the table space is ASCII or Unicode, you should specify the values as hexadecimal constants.
If more than one row has the value *literal* in the key column, REPAIR returns a list of record identifiers (RIDs) for records with that key value, but does not perform any other operations (verify, replace, delete, or dump) until the next LOCATE TABLESPACE statement is encountered. To repair the proper data, write a LOCATE TABLESPACE statement that selects the row that you want, using the RID option, the PAGE option, or a different KEY and INDEX option. Then, execute REPAIR again.

**SHRLEVEL**

Indicates the type of access that is to be allowed for the index, table space, or partition that is to be repaired during REPAIR processing.

If you do not specify SHRLEVEL and you do specify DUMP or VERIFY, applications can read but not write the area.

If you do not specify SHRLEVEL and you do specify DELETE or REPLACE, applications cannot read or write the area.

**CHANGE**

Specifies that applications can read and write during the VERIFY, REPLACE, DELETE, and DUMP operation.

**ROWID X'byte-string'**

Specifies that the data that is to be located is a LOB in a LOB table space.

*byte-string* is the row ID that identifies the LOB column.

Use the ROWID keyword to repair an orphaned LOB row. You can find the ROWID in the output from the CHECK LOB utility. If you specify the ROWID keyword, the specified table space must be a LOB table space.

**VERSION X'byte-string'**

Specifies that the data that is to be located is a LOB in a LOB table space.

*byte-string* is the version number that identifies the version of the LOB column.

Use the VERSION keyword to repair an orphaned LOB column. You can find the VERSION number in the output of the CHECK LOB utility or an out-of-synch LOB that is reported by the CHECK DATA utility. If you specify the VERSION keyword, the specified table space must be a LOB table space.

**LOCATE block: LOCATE INDEX statement and LOCATE INDEXSPACE statement option descriptions**

The LOCATE INDEX (or INDEXSPACE) statement locates data that is to be repaired within an index. You can specify indexes by either their index name or their index space name.

One LOCATE statement is required for each unit of data that is to be repaired. Multiple LOCATE statements can appear after each REPAIR statement.

If a REPAIR statement is followed by multiple LOCATE statements, all processing that is caused by VERIFY, REPLACE, and DUMP statements is committed before the next LOCATE statement is processed.

**INDEX index-name**

Specifies a particular index that is to be used to find the row that contains the key. When you are locating an index by key, the index that you specify must be a single-column index.
*index-name* is the qualified or unqualified name of the index. If you omit the qualifier creator ID, the user identifier for the utility job is used. Enclose the index name in quotation marks if the name contains a blank.

**INDEXSPACE**  *index-space-name*

Specifies the index space for a particular index that is to be used to find the row that contains the key. Look in the SYSEIM.SYSINDEXES catalog table to find the index space name for an index. When you are locating an index by key, the index that you specify must be a single-column index.

*index-space-name* is the qualified name of the index space, in the form `database-name.index-space-name`.

**PAGE**  *integer*

Specifies the relative page number within the index space that is to be operated on. The first page is 0 (zero).

*integer*

*integer* is a decimal number from one to six digits in length.

*X'byte-string'*

Specifies that the data of interest is an entire page. The specified offsets in *byte-string* and in subsequent statements are relative to the beginning of the page. The first byte of the page is at offset 0.

*byte-string* is a hexadecimal value from one to eight characters in length. You do not need to enter leading zeros. Enclose the *byte-string* between apostrophes, and precede it with X.

**PART**  *integer*

Specifies the partition number of the partitioning index that contains the page that is to be located. The PART keyword is valid only for indexes of partitioned table spaces.

*integer*

*integer* is the number of the partitioning index.

**LOCATE block: VERIFY statement**

The VERIFY statement tests whether a particular data area contains a specified value. Depending on the outcome of this test, the REPAIR utility performs the following actions:

- If the data area does contain the value, subsequent operations in the same LOCATE block are allowed to proceed.
- If any data area does not contain its specified value, all subsequent operations in the same LOCATE block are inhibited.

**LOCATE block: VERIFY statement option descriptions**

**OFFSET**

Locates the data that is to be tested by a relative byte address (RBA) within the row or page.

*integer*

*integer* identifies the offset as an integer.

The default value is 0, the first byte of the area that is identified by the previous LOCATE statement.
X'\textit{byte-string}'
Identifies the offset as one to four hexadecimal characters. You do not need to enter leading zeros. Enclose the byte string between apostrophes, and precede it with X.

\textbf{DATA}
Specifies what data must be present at the current location before a change is made.

Character constants that are specified within the VERIFY DATA option cannot be specified as ASCII or Unicode character strings. No conversion of the values is performed. To use this option when the table space is ASCII or Unicode, you should specify the values as hexadecimal constants.

X'\textit{byte-string}'
Specifies an even number, from 2 to 32, of hexadecimal characters that must be present. You do not need to enter leading zeros. Enclose the byte string between apostrophes, and precede it with X.

'\textit{character-string}'
Specifies any character string that must be present.

\textbf{LOCATE block: REPLACE statement}

The REPLACE statement replaces data at a particular location. The statement is contained within a LOCATE block. If any VERIFY statement within that block finds a data area that does not contain its specified data, the REPLACE operation is inhibited.

\textbf{LOCATE block: REPLACE statement option descriptions}

\textbf{RESET}
Specifies that the inconsistent data indicator is to be reset. A page for which this indicator is on is considered in error, and the indicator must be reset before you can access the page. Numbers of pages with inconsistent data are reported at the time that they are encountered.

The option also resets the PGCOMB flag bit in the first byte of the page to agree with the bit code in the last byte of the page.

\textbf{OFFSET}
Indicates where data is to be replaced by a relative byte address (RBA) within the row or page. Only one OFFSET and one DATA specification are acted on for each REPLACE statement.

\textit{integer}
Specifies the offset as an integer.

The default value is 0, the first byte of the area that is identified by the previous LOCATE statement.

X'\textit{byte-string}'
Specifies the offset as one to four hexadecimal characters. You do not need to enter leading zeros. Enclose the byte string between apostrophes, and precede it with X.

\textbf{DATA}
Specifies the new data that is to be entered. Only one OFFSET and one DATA specification are acted on for each REPLACE statement.
**Important:** Do not run REPAIR with the REPLACE, OFFSET, and DATA options on a compressed table space.

Character constants that are specified within the VERIFY DATA option cannot be specified as ASCII or Unicode character strings. The values are not converted. To use this option when the table space is ASCII or Unicode, specify the values as hexadecimal constants.

- **X’byte-string’**
  Specifies an even number, from 2 to 32, of hexadecimal characters that are to replace the current data. You do not need to enter leading zeros. Enclose the byte string between apostrophes, and precede it with X.

- **’character-string’**
  Specifies any character string that is to replace the current data.

**LOCATE block: DELETE statement**

The DELETE statement deletes a single row of data that has been located by a RID or KEY option. The statement is contained within a LOCATE block. If any VERIFY statement within that block finds a data area that does not contain its specified data, the DELETE operation is inhibited.

The DELETE statement operates without regard for referential constraints. If you delete a parent row, its dependent rows remain unchanged in the table space.

In any LOCATE block, you can include no more than one DELETE option.

If you have coded any of the following options, you cannot use DELETE:
- The LOG NO option on the REPAIR statement
- A LOCATE INDEX statement to begin the LOCATE block
- The PAGE option on the LOCATE TABLESPACE statement in the same LOCATE block
- A REPLACE statement for the same row of data

When you specify LOCATE ROWID for a LOB table space, the LOB that is specified by ROWID is deleted with its index entry. All pages that are occupied by the LOB are converted to free space. The DELETE statement does not remove any reference to the deleted LOB from the base table space.

When you specify LOCATE DOCID for an XML table space, the XML document that is specified by DOCID is deleted with its NodeID index entries. All rows that are occupied by the XML document are deleted from the XML table space. The DELETE statement does not remove any reference to the deleted XML document from the base table space. The LOCATE DOCID statement is generated by CHECK DATA SHRLEVEL CHANGE in order to remove corrupted XML documents from the XML table space.

REPAIR DELETE can delete the following data rows when the specified conditions exist:
- A compressed row without an index defined on the table
- A compressed row with an index defined on the table and a valid dictionary exists to decompress the row
- A compressed or uncompressed data row that is missing an index entry
• A compressed row with an index defined on the table, but the dictionary is invalid
• An uncompressed row without an index
• An uncompressed row with valid Index

**LOCATE block: DELETE statement option descriptions**

**DATAONLY**
- Specifies that REPAIR should delete only the data record that is specified by the LOCATE RID statement. Any associated indexes, LOB columns, XML columns, or referential integrity constraints are not deleted.

  You can specify the DATAONLY option only when REPAIR locates a single row by using a RID.

  If the table has indexes or LOB or XML columns, ensure that after you run the DELETE DATAONLY statement, the data is consistent with the other associated objects.

**LOCATE block: DUMP statement**

The DUMP statement produces a hexadecimal dump of data that is identified by offset and length. DUMP statements have no effect on VERIFY or REPLACE operations.

When you specify LOCATE ROWID for a LOB table space, one or more map or data pages of the LOB are dumped. The DUMP statement dumps all of the LOB column pages if you do not specify either the MAP or DATA keyword.

**LOCATE block: DUMP statement option descriptions**

**OFFSET**
- Optionally, locates the data that is to be dumped by a relative byte address (RBA) within the row or page.

  integer
  - Specifies the offset as an integer.
  - The default value is 0, the first byte of the row or page.

  X'byte-string'
  - Specifies the offset as one to four hexadecimal characters. You do not need to enter leading zeros. Enclose the byte string between apostrophes, and precede it with X.

**LENGTH**
- Optionally, specifies the number of bytes of data that are to be dumped. If you omit both LENGTH and PAGE, the dump begins at the specified OFFSET and continues to the end of the row or page.

  If you specify a number of bytes (with LENGTH) and a number of pages (with PAGE), the dump contains the same relative bytes from each page. That is, from each page you see the same number of bytes, beginning at the same offset.

  X'byte-string'
  - Specifies one to four hexadecimal characters. You do not need to enter leading zeros. Enclose the byte string between apostrophes, and precede it with X.
integer
   Specifies the length as an integer.

PAGES
   Optionally, specifies a number of pages that are to be dumped. You can use
   this option only if you used PAGE in the preceding LOCATE TABLESPACE
   control statement.

X'byte-string'
   Specifies one to four hexadecimal characters. You do not need to enter
   leading zeros. Enclose the byte string between apostrophes, and precede it
   with X.

integer
   Specifies the number of pages as an integer.

*  Specifies that all pages from the starting point to the end of the table space
   or partition are to be dumped.

MAP pages
   Specifies that only the LOB map pages are to be dumped.

   pages specifies the number of LOB map pages that are to be dumped. If you do
   not specify pages, all LOB map pages of the LOB that is specified by ROWID
   and version are dumped.

DATA pages
   Specifies that only the LOB data pages are to be dumped.

   pages specifies the number of LOB data pages that are to be dumped. If you do
   not specify pages, all LOB data pages of the LOB that is specified by ROWID
   and version are dumped.

SET statement

The SET TABLESPACE statement resets the COPY-pending, RECOVER-pending,
CHECK-pending, auxiliary warning (AUXW), auxiliary CHECK-pending
(ACHKP), and advisory REORG-pending (AREO* and AREOR) statuses for a table
space or data set. The SET TABLESPACE statement also turns on and off Persistent
Read Only (PRO) restricted status for a table space partition. The SET INDEX
statement resets the informational COPY-pending (ICOPY), RECOVER-pending,
REBUILD-pending, CHECK-pending, and advisory REORG-pending (AREO* and
AREOR) statuses for an index.

If you do not specify a status to reset, REPAIR takes no action.

SET statement option descriptions

SET TABLESPACE database-name.table-space-name
   Specifies the table space (and, optionally, the database to which it belongs)
   whose pending status is to be reset.

database-name
   Specifies the name of the database to which the table space belongs.

   The default value is DSNDB04.

table-space-name
   Specifies the name of the table space.
SET INDEX
Specifies the index whose RECOVER-pending, CHECK-pending, REBUILD-pending, or informational COPY-pending status is to be reset.

(index-name)
Specifies the index that is to be processed. Enclose the index name in quotation marks if the name contains a blank.

(ALL)
Specifies that all indexes in the table space will be processed.

You can also repair all indexes by specifying INDEX(ALL) followed by a table-space-spec.

SET INDEXSPACE
Specifies the index space for the index whose RECOVER-pending, CHECK-pending, REBUILD-pending, or informational COPY-pending status is to be reset.

(database-name.index-space-name)
Specifies the index space that is to be processed.

(ALL)
Specifies that all indexes in the table space will be processed.

PART integer
Specifies a particular partition whose COPY-pending, or RECOVER-pending status is to be reset. If you do not specify PART, REPAIR resets the pending status of the entire table space or index.

integer is the number of the partition and must be in the range from one to the number of partitions that are defined for the

You can specify PART for NOCHECKPEND on a table space, and for NORCVRPEND on indexes.

The PART keyword is not valid for a LOB table space or an index on the auxiliary table.

The PART keyword is not valid when NOAREORPEND is specified because the AREOR state can only be reset for the entire table space or index space.

NOCOPYPEND
Specifies that the COPY-pending status of the specified table space, or the informational COPY-pending (ICOPY) status of the specified index is to be reset.

NORCVRPEND
Specifies that the RECOVER-pending (RECP) status of the specified table space or index is to be reset.

NORBDPEND
Specifies that the REBUILD-pending (RBDP) status, the page set REBUILD-pending status (PSRBDP), or the RBDP* status of the specified index is to be reset.

NOCHECKPEND
Specifies that the CHECK-pending (CHKP) status of the specified table space or index is to be reset.
NOAUXWARN
Specifies that the auxiliary warning (AUXW) status of the specified table space is to be reset. The specified table space must be a base table space or a LOB table space.

NOAUXCHKP
Specifies that the auxiliary CHECK-pending (ACHKP) status of the specified table space is to be reset. The specified table space must be a base table space.

NOAREORPENDSTAR
Resets the advisory REORG-pending (AREO*) status of the specified table space or index.

NOAREORPEND
Resets the advisory REORG-pending (AREOR) status of the specified table space or index.

PRO
Turns on Persistent Read Only (PRO) restricted status for a table space partition. The PART keyword is required for this option.

NOPRO
Turns off Persistent Read Only (PRO) restricted status for a table space partition. The PART keyword is required for this option.

RBDPEND
Specifies that the REBUILD-pending (RBDP) status is to be set on the specified index.

PSRBDPEND
Specifies that the PAGE SET REBUILD-pending (PSRBDP) status is to be set on the specified index.

**DBD statement**

Use the DBD statement to perform one or more of the following actions:
- Compare the database definition (DBD) in the Db2 catalog with its definition in the Db2 directory
- Rebuild a database definition in the directory by using the information, including LOB information, in the Db2 catalog
- Drop an inconsistent database definition from the Db2 catalog and the Db2 directory

For more information about how to use the DBD statement to perform these actions, see [“Repairing DBDs” on page 696](#).

**DBD statement option descriptions**

DROP
Specifies that the named database is to be dropped from both the Db2 catalog and the Db2 directory. When you specify this option, Db2 also drops databases that contain tables that have been created with RESTRICT ON DROP. Use this keyword if the SQL DROP DATABASE statement fails because the description of the database is not in both the Db2 catalog and the Db2 directory. If you cannot use the ALTER command to remove the with RESTRICT ON DROP option on tables in a database that is badly damaged and you need to drop the database, you can use this keyword to drop the database.
**Attention:** Use the DROP option with extreme care. Using DROP can cause additional damage to your data. For more assistance, you can contact IBM Software Support.

**DATABASE database-name**
Specifies the target database.

*database-name* is the name of the target database, which cannot be DSNDB01 (the Db2 directory) or DSNDB06 (the Db2 catalog).

If you use TEST, DIAGNOSE, or REBUILD, *database-name* cannot be DSNDB07 (the work file database).

If you use DROP, *database-name* cannot be DSNDB04 (the default database).

**DBID X'\text{dbid}'**
Specifies the database descriptor identifier for the target database.

*dbid* is the database descriptor identifier.

**TEST**
Specifies that a DBD is to be built from information in the Db2 catalog, and is to be compared with the DBD in the Db2 directory. If you specify TEST, Db2 reports significant differences between the two DBDs.

If the condition code is 0, the DBD in the Db2 directory is consistent with the information in the Db2 catalog.

If the condition code is not 0, then the information in the Db2 catalog and the DBD in the Db2 directory might be inconsistent. Run REPAIR DBD with the DIAGNOSE option to gather information that is necessary for resolving any possible inconsistency.

**DIAGNOSE**
Specifies that information that is necessary for resolving an inconsistent database definition is to be generated. Like the TEST option, DIAGNOSE builds a DBD that is based on the information in the Db2 catalog and compares it with the DBD in the Db2 directory. In addition, Db2 reports any differences between the two DBDs, and produces hexadecimal dumps of the inconsistent DBDs.

If the condition code is 0, the information in the Db2 catalog and the DBD in the Db2 directory is consistent.

If the condition code is 8, the information in the Db2 catalog and the DBD in the Db2 directory might be inconsistent.

For further assistance in resolving any inconsistencies, you can contact IBM Support.

**REBUILD**
Specifies that the DBD that is associated with the specified database is to be rebuilt from the information in the Db2 catalog.

**Attention:** Use the REBUILD option with extreme care, as you can cause more damage to your data. For more assistance, you can contact IBM Software Support.

**OUTDDN ddname**
Specifies the DD statement for an optional output data set. This data set contains copies of the Db2 catalog records that are used to rebuild the DBD.

*ddname* is the name of the DD statement.
LEVELID statement

Use the LEVELID statement to accept a down-level page set by changing its level ID. You cannot run REPAIR with any other REPAIR utility control statement.

Important: Accepting the use of a down-level data set might cause data inconsistencies. Problems with inconsistent data that result from resetting the level identifier are the responsibility of the user.

LEVELID statement option descriptions

LEVELID
Indicates that the level identifier of the named table space, table space partition, index, or index space partition is to be reset to a new identifier. Use LEVELID to accept the use of a down-level data set. You cannot specify multiple LEVELID keywords in the same REPAIR control statement.

You cannot use LEVELID with a table space, table space partition, index, or index space partition that has outstanding indoubt log records or pages in the logical page list (LPL).

TABLESPACE database-name.table-space-name
Specifies the table space whose level identifier is to be reset.

database-name
Specifies the name of the database to which the table space belongs.

The default value is DSND04.

table-space-name
Specifies the name of the table space.

INDEX
Specifies the index whose level identifier is to be reset.

creator-id
Specifies the creator of the index. Specifying this qualifier is optional.

index-name
Specifies the name of the index. Enclose the index name in quotation marks if the name contains a blank.

You can specify either INDEX or INDEXSPACE to identify an index. To specify multiple indexes, repeat the keyword.

INDEXSPACE
Specifies the index space for the index whose level identifier is to be reset. You can obtain the index space name for an index from the SYSIBM.SYSINDEXES catalog table. The index space name must be qualified.

database-name
Specifies the name of the database to which the index space belongs.

index-space-name
Specifies the name of the index space.

PART
Identifies a partition of the table space or index (including a partition of a data-partitioned secondary index).

integer is the physical partition number. It must be in the range from 1 to the number of partitions that are defined for the table space. The maximum is 4096.
VERSIONS statement

Use the VERSIONS statement to update version information in the catalog and directory for a table space or index with the version information from the system pages of the object. Use REPAIR VERSIONS only after you run the DSN1COPY utility with the OBIDXLAT option to move objects from one Db2 subsystem to another.

Important: The VERSIONS option is deprecated. Use repair REPAIR CATALOG instead.

VERSIONS statement option descriptions

VERSIONS
  Specifies that REPAIR updates the version information in the Db2 catalog or directory for a table space or index with the version information from the system pages of the object.

TABLESPACE database-name.table-space-name
  Specifies the table space whose version information is to be updated.
  
  database-name
    Specifies the name of the database to which the table space belongs.
    The default value is DSNDB04.
  
  table-space-name
    Specifies the name of the table space.

INDEX
  Specifies the index whose version information is to be updated.
  
  creator-id
    Specifies the creator of the index. Specifying this qualifier is optional.
  
  index-name
    Specifies the name of the index. Enclose the index name in quotation marks if the name contains a blank.

You can specify either INDEX or INDEXSPACE to identify an index. To specify multiple indexes, repeat the keyword.

INDEXSPACE
  Specifies the index space for the index whose level identifier is to be reset. You can obtain the index space name for an index from the SYSIBM.SYSINDEXES catalog table. The index space name must be qualified.
  
  database-name
    Specifies the name of the database to which the index space belongs.
  
  index-space-name
    Specifies the name of the index space.

PART
  Identifies a partition of the table space or index (including a partition of a data-partitioned secondary index).
  
  integer is the physical partition number. It must be in the range from 1 to the number of partitions that are defined for the table space. The maximum is 4096.
CATALOG statement

Use the CATALOG statement to check and correct inconsistencies between the catalog and a table space after DSN1COPY is used to create a copy of the object a different Db2 subsystem from the subsystem on which the original table space resides.

When you specify REPAIR CATALOG, the utility performs the following actions:

- Compares the following information in the catalog with the data:
  
  For the following items, if the information in the catalog is different from the data, REPAIR changes the values in the catalog to match the data:
  
  - Row format (Row format can be either reordered row format or basic row format.)
  - RBA format (RBA format can be either 6-byte format or 10-byte format.)
  - Data version information (This check is the same function that is performed when you specify REPAIR VERSIONS.)
  - Hash space value

  For the following items, if the information in the catalog is different from the data, REPAIR determines whether conversion from the page set format to the catalog format is supported. Supported conversions are any conversions that can be performed through ALTER TABLE SET DATA TYPE. If the conversion is supported, REPAIR increments the table space version in the catalog. This action causes the column data in the page set to be converted to the data type and length in the catalog definition the next time that the data is accessed. If conversion is not supported, an error is issued.
  
  - Column data types
  - Column lengths
  - Number of columns

  For this case, the number of the columns in the catalog definition of the table must be greater than the number of columns in the page set. All columns in the page set must have types and lengths that can be converted to the types and lengths of columns in the catalog.

- Validates the following information:

  For these items, if the information in the catalog is different from the data, REPAIR CATALOG does not correct the information in the catalog. Instead, REPAIR fails and reports the mismatched information in a message. To correct the mismatched information, take the action that is documented for the message that you receive.
  
  - DBID, PSID, and OBID
  - Table space type
  - SEGSIZE
  - PAGESIZE
  - Table definition

  - Whether the system page exists in the page set
  - Whether a table version number wrapped from 255 to 1
  - Whether a table is marked as dropped in the database descriptor, but is still defined in the catalog
  - The number of columns in the catalog definition of a table is fewer than the number of the columns in the page set
Updates the CURRENT_VERSION or OLDEST_VERSION column of the SYSIBM.SYSTABLESPACE table when the data in the table space is at different versions from the versions that are recorded in the catalog.

REPAIR CATALOG does not check limit key values.

REPAIR CATALOG does not make any corrections for indexes. If REPAIR or you made corrections to the data or catalog as a result of running REPAIR CATALOG, rebuild any indexes on the target tables.

**CATALOG statement option descriptions**

**CATALOG**

Specifies that REPAIR is to validate information in the catalog for the specified object.

You cannot specify CATALOG for LOB or XML table spaces.

**TABLESPACE database-name.table-space-name**

Specifies the table space for which catalog information is to be validated.

  *database-name*

  Specifies the name of the database to which the table space belongs.

  The default value is **DSNDB04**.

  *table-space-name*

  Specifies the name of the table space.

**TEST**

Indicates that REPAIR is not to correct any mismatched information. The utility checks all of the same information that it checks when you specify REPAIR CATALOG. However, any information differences between the data and catalog are only reported in messages. The utility does not take any corrective actions.

**System pages statement**

Use the system pages statement to insert missing system pages into the specified table space for tables that are in version 0 format. Insertion of system pages in version 0 format ensures that a table space or table is self-describing. That is, Db2 requires no catalog or directory information to determine the format of the table space or tables in the table space. Specification of this option is important for read-only objects that might be moved or copied to other Db2 subsystems.

Before running the system pages statement, run REPAIR CATALOG TEST to determine whether system pages need to be inserted. If REPAIR CATALOG TEST returns the message **MISSING SYSTEM PAGE IN PAGESET**, the system pages statement needs to be run.

The system pages statement cannot run on a table space that was the target of DSN1COPY, because REPAIR cannot determine whether the format of the data matches its format in the Db2 catalog.

**System pages statement option descriptions**

**INSERTVERSIONPAGES**

Indicates that REPAIR inserts missing system pages into the table space for tables that are in version 0 format.
SETCURRENTVERSION

Indicates that REPAIR takes the following actions:

- Inserts missing system pages for table spaces or tables in the table space that are in version 0 format.
- Updates the version numbers in the VERSION column of the SYSIBM.SYSTABLES catalog table for all tables in the table space that are at version 0 to the same version number as in the CURRENT_VERSION column in SYSIBM.SYSTABLESPACE.

After running REPAIR with INSERTVERSIONPAGES and SETCURRENTVERSION, run REORG TABLESPACE to update the data rows to the current table space version. Then run MODIFY RECOVERY to remove old versions.

SHRLEVEL CHANGE

Indicates the type of access that is to be allowed for the table space into which system pages are inserted.

If you do not specify SHRLEVEL CHANGE, applications can read but not write in the table space.

If you specify SHRLEVEL CHANGE, you cannot specify SETCURRENTVERSION.

TABLESPACE database-name.table-space-name

Specifies the table space whose version information is to be updated.

database-name

Specifies the name of the database to which the table space belongs.

The default value is **DSNDB04**.

table-space-name

Specifies the name of the table space.

Related tasks:

"Copying tables from one subsystem to another" on page 981

---

**Before running REPAIR**

Certain activities might be required before you run the REPAIR utility, depending on your situation.

**Attention:** Be extremely careful when using the REPAIR utility to replace data. Changing data to invalid values by using REPLACE might produce unpredictable results, particularly when changing page header information. Improper use of REPAIR can result in damaged data, or in some cases, system failure.

**Making a copy of the table space**

Before starting to use REPAIR to change data, ensure that you have a copy (full image copy or DSN1COPY generated copy) of the affected table space to enable fallback.

**Restoring damaged indexes**

Because REPAIR can access index data only by referring to a page and an offset within the page, identifying and correcting a problem can be difficult. Use REBUILD INDEX or RECOVER INDEX to restore damaged index data.
Running REPAIR on encrypted data

Do not run REPAIR on encrypted data. REPAIR does not decrypt the data. The utility reads the data in its encrypted form and then manipulates the data without decrypting it.

Data sets that REPAIR uses

The REPAIR utility uses a number of data sets during its operation.

The following table lists the data sets that REPAIR uses. The table lists the DD name that is used to identify the data set, a description of the data set, and an indication of whether it is required. Include statements in your JCL for each required data set and any optional data sets that you want to use.

Table 99. Data sets that REPAIR uses

<table>
<thead>
<tr>
<th>Data set</th>
<th>Data set description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSIN</td>
<td>Input data set that contains the utility control statement.</td>
<td>Yes</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>Output data set for messages.</td>
<td>Yes</td>
</tr>
<tr>
<td>Optional output data set</td>
<td>Data set that contains copies of the Db2 catalog records that are used to rebuild the DBD. You define the DD name.</td>
<td>No</td>
</tr>
</tbody>
</table>

The following objects are named in the utility control statement and do not require a DD statement in the JCL:

Table space or index

Object that is to be repaired.

Calculating output data set size

Use the following formula to estimate the size of the output data set:

\[
\text{SPACE} = (4096, (n,n))
\]

In this formula, \(n\) = the total number of Db2 catalog records that relate to the database on which REPAIR DBD is being executed.

You can calculate an estimate for \(n\) by summing the results of SELECT COUNT(*) from the following catalog tables, for catalog table rows whose database name matches the name of the database on which REPAIR DBD is being executed.

- Syscolauth
- Syscolumns
- Sysfields
- Sysforeignkeys
- Sysindexes
- Sysindexpart
- Syskeys
- Sysrels
- Sys synonymys
- Systabauth
- Systablepart
- Sytables
- Sytablespace

Related concepts:
Concurrency and compatibility for REPAIR

The REPAIR utility has certain concurrency and compatibility characteristics associated with it.

Db2 treats individual data and index partitions as distinct target objects. Utilities that operate on different partitions of the same table space or index space are compatible.

Claims

The following table shows which claim classes REPAIR drains and any restrictive state that the utility sets on the target object.

Table 100. Claim classes of REPAIR operations

<table>
<thead>
<tr>
<th>Action</th>
<th>Table space or partition</th>
<th>Index or partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPAIR LOCATE KEY DUMP or VERIFY</td>
<td>DW/UTRO</td>
<td>DW/UTRO</td>
</tr>
<tr>
<td>REPAIR LOCATE KEY DELETE or REPLACE</td>
<td>DA/UTUT</td>
<td>DA/UTUT</td>
</tr>
<tr>
<td>REPAIR LOCATE RID DUMP or VERIFY</td>
<td>DW/UTRO</td>
<td>None</td>
</tr>
<tr>
<td>REPAIR LOCATE RID DELETE</td>
<td>DA/UTUT</td>
<td>DA/UTUT</td>
</tr>
<tr>
<td>REPAIR LOCATE RID REPLACE</td>
<td>DA/UTUT</td>
<td>None</td>
</tr>
<tr>
<td>REPAIR LOCATE TABLESPACE DUMP or VERIFY</td>
<td>DW/UTRO</td>
<td>None</td>
</tr>
<tr>
<td>REPAIR LOCATE TABLESPACE REPLACE</td>
<td>DA/UTUT</td>
<td>None</td>
</tr>
<tr>
<td>REPAIR LOCATE INDEX PAGE DUMP or VERIFY</td>
<td>None</td>
<td>DW/UTRO</td>
</tr>
<tr>
<td>REPAIR LOCATE INDEX PAGE DELETE</td>
<td>None</td>
<td>DA/UTUT</td>
</tr>
</tbody>
</table>

Legend:
- DA - Drain all claim classes - no concurrent SQL access.
- DW - Drain the write claim class - concurrent access for SQL readers.
- UTUT - Utility restrictive state - exclusive control.
- UTRO - Utility restrictive state - read-only access allowed.
- None - Object is not affected by this utility.

REPAIR does not set a utility restrictive state if the target object is DSNDB01.SYSUTILX.

Compatibility

The following tables show which utilities can run concurrently with REPAIR on the same target object. The target object can be a table space, an index space, or a partition of a table space or index space. If compatibility depends on particular options of a utility, that information is also shown in the table.

Note that REPAIR SHRLEVEL CHANGE obtains an exclusive mass delete lock on a segmented table space, which prevents execution of any application for which both of the following conditions are true:
- The application is bound with ISO(UR).
• The application issues a statement (such as a SELECT) that acquires a mass delete lock on that table space.

The following table shows which utilities can run concurrently with REPAIR LOCATE by KEY or RID.

Table 101. Utility compatibility with REPAIR, LOCATE by KEY or RID

<table>
<thead>
<tr>
<th>Utility</th>
<th>DUMP or VERIFY</th>
<th>DELETE or REPLACE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHECK DATA</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CHECK INDEX</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>CHECK LOB</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>COPY INDEXSPACE</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>COPY TABLESPACE</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>DIAGNOSE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LOAD</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MERGECOPY</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MODIFY</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>QUIESCE</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>REBUILD INDEX</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RECOVER INDEX</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RECOVER TABLESPACE</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REORG INDEX</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REORG TABLESPACE UNLOAD</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REORG TABLESPACE UNLOAD ONLY or EXTERNAL</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>REPAIR DELETE or REPLACE</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REPAIR DUMP or VERIFY</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>REPORT</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RUNSTATS INDEX SHRLEVEL CHANGE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Notes:
1. REORG INDEX is compatible with LOCATE by RID, DUMP, VERIFY, or REPLACE.
2. RECOVER INDEX is compatible with LOCATE by RID, DUMP, or VERIFY.
3. REPAIR LOCATE INDEX PAGE REPLACE is compatible with LOCATE by RID or REPLACE.

The following table shows which utilities can run concurrently with REPAIR LOCATE by PAGE.
Table 102. Utility compatibility with REPAIR, LOCATE by PAGE

<table>
<thead>
<tr>
<th>Utility or action</th>
<th>TABLESPACE DUMP or VERIFY</th>
<th>TABLESPACE REPLACE</th>
<th>INDEX DUMP or VERIFY</th>
<th>INDEX REPLACE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL read</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>SQL write</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CHECK DATA</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CHECK INDEX</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>CHECK LOB</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>COPY INDEXSPACE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>COPY TABLESPACE</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>DIAGNOSE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LOAD</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MERGECOPY</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MODIFY</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>QUIESCE</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>REBUILD INDEX</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>RECOVER INDEX</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RECOVER TABLESPACE</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>(with no option)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECOVER TABLESPACE</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ERROR RANGE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECOVER TABLESPACE</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>TOCOPY or TORBA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REORG INDEX</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REORG TABLESPACE</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>UNLOAD CONTINUE or</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAUSE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REORG TABLESPACE</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UNLOAD ONLY or EXTERNAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REPAIR DELETE or</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REPLACE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REPAIR DUMP or VERIFY</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>REPORT</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RUNSTATS INDEX</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>RUNSTATS TABLESPACE</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>STOSPACE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UNLOAD</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Note:
1. REPAIR LOCATE INDEX PAGE REPLACE is compatible with LOCATE TABLESPACE PAGE.

Resetting table space status

In most cases, resetting the COPY-pending restriction by taking a full image copy is preferable to using REPAIR. This is because RECOVER cannot be executed successfully until an image copy has been made.
Resetting the RECOVER-pending status by running RECOVER or LOAD is preferable to using REPAIR. This is because RECOVER uses Db2-controlled recovery information, whereas REPAIR SET TABLESPACE or INDEX resets the RECOVER-pending status without considering the recoverability of the table space. Recoverability issues include the availability of image copies, of rows in SYSIBM.SYSCOPY, and of log data sets.

Verifying and possibly correcting referential integrity constraints by running CHECK DATA are recommended. CHECK DATA performs a complete check of all referential integrity constraints of the table space set, whereas with REPAIR, you are responsible for checking all the referential integrity constraints violations.

To reset the CHECK-pending status for a LOB table space:
1. Run the CHECK DATA utility again with the AUXERROR INVALIDATE keywords specified.
2. Update the invalid LOBs.

To reset the auxiliary warning (AUXW) status for a LOB table space:
1. Update or correct the invalid LOB columns, then
2. Run the CHECK LOB utility with the AUXERROR INVALIDATE option if invalid LOB columns were corrected.

**Resetting index space status**

Running COPY INDEXSPACE to reset the informational COPY-pending status is preferable to using the REPAIR utility to reset the status.

Consider using the REBUILD INDEX or RECOVER INDEX utility on an index that is in REBUILD-pending status, rather than running REPAIR SET INDEX NORDPEND. RECOVER uses Db2-controlled recovery information, whereas REPAIR SET INDEX resets the REBUILD-pending status without considering the recoverability of the index. Recoverability issues include the availability of image copies, of rows in SYSIBM.SYSCOPY, and of log data sets.

**Repairing a damaged page**

You can use the REPAIR utility to repair a damaged page.

**Procedure**

To repair a damaged page:
1. Execute REPAIR with the LOG YES option and the DUMP control statement, specifying the pages that you suspect are damaged. Then, verify that the dump you received contains the pages that you want.
2. If you know which page is damaged and you can see how to resolve the error, repair the page and reset the “inconsistent data” indicator. Run REPAIR with the REPLACE RESET DATA control statement. Document your actions in case you need to undo anything later.
3. If you determine that the page is not really damaged, but merely has the “inconsistent data” indicator on, reset the indicator by running REPAIR with the REPLACE RESET control statement.
Repairing DBDs

You can check and repair database definitions (DBDs) in the catalog and directory by using the REPAIR utility with the DBD statement.

About this task

You can use REPAIR DBD on declared temporary tables, which must be created in a database that is defined with the AS TEMP clause. No other Db2 utilities can be used on a declared temporary table, its indexes, or its table spaces.

Procedure

To repair DBDs:
1. Run REPAIR DBD with the TEST option to determine whether the information in the Db2 catalog is consistent with the DBD in the Db2 directory.
   REPAIR DBD TEST obtains environment information, such as the character that is used for the decimal point, from the application defaults load module that is used by the subsystem. The application defaults load module is either the default load module DSNHDECP or a user-specified application defaults load module. If the return code is not 0, inconsistencies exist.
2. If inconsistencies exist, run REPAIR DBD with the DIAGNOSE and OUTDDN options to produce diagnostic information.
   REPAIR DBD DIAGNOSE obtains environment information, such as the character that is used for the decimal point, from the application defaults load module that is used by the subsystem.
   Contact IBM Support for assistance in analyzing this information.
3. If IBM Support instructs you to do so, replace the existing DBD by running REPAIR DBD with the REBUILD option.
   Attention: Do not use the REBUILD option if you suspect that information in the catalog is causing the inconsistency. REBUILD uses information in the catalog to rebuild the DBD; if the catalog is incorrect, the rebuilt DBD cannot be correct.
   REPAIR DBD REBUILD obtains environment information, such as the character that is used for the decimal point, from the DSNHDECP module for the subsystem.
   Db2 starts the database for access by utilities only. After successful completion of the REPAIR utility, the database continues to be started for utility access only.
   When REPAIR DBD REBUILD is running, an S-lock is acquired for the appropriate catalog tables. If the S-lock fails, REPAIR DBD fails.
   Db2 reads each table space in the database during the REBUILD process to gather information. If the data sets for the table spaces do not exist or are not accessible to Db2, the utility abnormally terminates.
4. If you suspect an inconsistency in the DBD of the work file database, consider issuing the DROP DATABASE SQL statement or running REPAIR DBD DROP. Then re-create the database.
   Attention: Use REPAIR DBD DROP with extreme care. Using DROP can cause additional damage to your data. For more assistance, contact IBM Support.
   If you receive errors when you drop the work file database, contact IBM Support for assistance.
5. If you ran REPAIR DBD REBUILD, the database is started for utility-only access, and you must restart the database for read/write access manually by performing the following steps:
   a. Issue the STOP DATABASE (database-name) command.
   b. Issue the START DATABASE (database-name) ACCESS(RW) command to allow full access to the database.

6. Rebind any trigger packages that were invalidated.
   When you run REPAIR DBD REBUILD on a database, Db2 invalidates packages for any triggers that are defined on tables in that database. To find those triggers, use the following query:
   ```sql
   SELECT T.NAME, T.SCHEMA FROM SYSIBM.SYSTRIGGERS T, SYSIBM.SYSDATABASE D WHERE T.DBID = D.DBID AND D.NAME = 'your database name here'
   ```
   After you run REPAIR DBD REBUILD, you must rebind those trigger packages. The Db2 release on which you rebind the trigger packages must be the same as the Db2 release on which you ran REPAIR DBD REBUILD.

Related reference:

- “Syntax and options of the REPAIR control statement” on page 670
- DROP (Db2 SQL)
- START DATABASE (Db2) (Db2 Commands)
- STOP DATABASE (Db2) (Db2 Commands)

Locating rows by key

If you use LOCATE TABLESPACE KEY, a number of rows might satisfy the condition. In this case, REPAIR returns only the RIDs of the rows and does not perform any VERIFY, REPLACE, DELETE, or DUMP actions which might be coded in that LOCATE block.

You can use the RID option of LOCATE TABLESPACE to identify a specific row. Examples of the messages that are issued are shown in the following example:

DSNU658I - DSNUCBRL - MULTIPLE RECORDS FOUND WITH SPECIFIED KEY
DSNU6601 - DSNUCBRL - POSSIBLE RID - X00000100B'
DSNU6601 - DSNUCBRL - POSSIBLE RID - X000000C18'
DSNU6601 - DSNUCBRL - POSSIBLE RID - X000000916'
DSNU6601 - DSNUCBRL - POSSIBLE RID - X000000513'
DSNU6501 - DSNUCBRP - DUMP
DSNU012I DSNUGBAC - UTILITY EXECUTION TERMINATED, HIGHEST RETURN CODE=8

Multiple-column indexes

The KEY option supports only single-column indexes. The following message is issued if you try to locate a row by using a multiple-column index.

DSNUCBRK - INDEX USED HAS MULTIPLE-FIELD KEY

Using VERIFY with REPLACE and DELETE operations

If any data area does not contain the value that is required by a VERIFY statement, all REPLACE and DELETE operations in the same locate block are inhibited. VERIFY and REPLACE statements that follow the next LOCATE statement are not affected.
Repairing critical catalog table spaces and indexes

An ID with a granted authority receives message DSNT500I RESOURCE UNAVAILABLE, while trying to repair a table space or index in the catalog or directory if table space DSNDB06.SYSUSER is unavailable.

About this task

If you get this message, you must either make these table spaces available or run the REPAIR utility on the catalog or directory by using an authorization ID with the installation SYSADM or installation SYSOPR authority.

Related information:

Termination or restart of REPAIR

You can terminate the REPAIR utility, but you cannot restart the REPAIR utility.

You can terminate a REPAIR job with the TERM UTILITY command.

REPAIR cannot be restarted. If you attempt to restart REPAIR, you receive message DSNU181I, which states that the utility cannot be restarted. You must terminate the job with the TERM UTILITY command, and rerun REPAIR from the beginning.

Related reference:

Review of REPAIR output

The output from the REPAIR utility can consist of any modified pages in the specified Db2 table space or index. Alternatively, the REPAIR utility can produce a complete dump of the content of the table space.

Error messages

At each LOCATE statement, the last data page and the new page that are being located are checked for a few common errors, and messages are issued.

Data checks

Although REPAIR enables you to manipulate both user and Db2 data by bypassing SQL, it does perform some checking of data. For example, if REPAIR tries to write a page with the wrong page number, Db2 abnormally terminates with a 04E code and reason code C200B0. If the page is broken because the broken page bit is on or the incomplete page flag is set, REPAIR issues the following message:

DSNU670I + DSNUCBRP - PAGE X'000004' IS A BROKEN PAGE

After running REPAIR

Certain activities might be required after you run the REPAIR utility, depending on your situation.
CHECK-pending status

You are responsible for violations of referential constraints that are a result of running REPAIR. These violations cause the target table space to be placed in the CHECK-pending status.

After running REPAIR DBD REBUILD

Make sure that you rebind any invalidated trigger packages. See the information about repairing DBDs.

Related tasks:
"Repairing DBDs” on page 696

Related reference:
Chapter 11, “CHECK DATA,” on page 73

Sample REPAIR control statements

Use the sample control statements as models for developing your own REPAIR control statements.

Example: Replacing damaged data and verifying replacement

The following control statement specifies that the REPAIR utility is to perform the following actions:

- Repair the specified page of table space DSN8D81A.DSN8S81D, as indicated by the LOCATE clause.
- Verify that, at the specified offset (50), the damaged data (0A00) is found, as indicated by the VERIFY clause.
- Replace the damaged data with the data that you want (0D11), as indicated by the REPLACE clause.
- Initiate a dump beginning at offset 50, for 4 bytes, as indicated by the DUMP clause. You can use the generated dump to verify the replacement.

```
//STEP1 EXEC DSNUPROC,UID='IUIQULH',UTPROC='',SYSTEM='DSN'
//SYSIN DD *
REPAIR OBJECT
  LOCATE TABLESPACE DSN8D81A.DSN8S81D PAGE X'02'
    VERIFY OFFSET 50 DATA X'0A00'
    REPLACE OFFSET 50 DATA X'0D11'
    DUMP OFFSET 50 LENGTH 4
```

Example: Removing a nonindexed row that is found by REORG

When reorganizing table space DSNDB04.TS1, assume that you received the following message:

```
DSNU3401 DSNURBXA - ERROR LOADING INDEX, DUPLICATE KEY
  INDEX = EMPIINDEX
  TABLE = EMP
  RID OF INDEXED ROW = X'0000000201'
  RID OF NONINDEXED ROW = X'0000000503'
```

To resolve this error condition, submit the following control statement, which specifies that REPAIR is to delete the nonindexed row and log the change. (The LOG keyword is not required; the change is logged by default.) The RID option identifies the row that REPAIR is to delete.
REPAIR
LOCATE TABLESPACE DSNDB04.TS1 RID (X'0000000503')
DELETE

Example: Reporting whether catalog and directory DBDs differ

The following control statement specifies that REPAIR is to compare the DBD for DSN8D2AP in the catalog with the DBD for DSN8D2AP in the directory.

REPAIR DBD TEST DATABASE DSN8D2AP

If the condition code is 0, the DBDs are consistent. If the condition code is not 0, the DBDs might be inconsistent. In this case, run REPAIR DBD with the DIAGNOSE option, as shown in example 4, to find out more detailed information about any inconsistencies.

Example: Reporting differences between catalog and directory DBDs

The following control statement specifies that the REPAIR utility is to report information about the inconsistencies between the catalog and directory DBDs for DSN8D2AP. Run this job after you run a REPAIR job with the TEST option (as shown in example 3), and the condition code is not 0. In this example, SYSREC is the output data set, as indicated by the OUTDDN option.

REPAIR DBD DIAGNOSE DATABASE DSN8D2AP OUTDDN SYSREC

Example: Resetting restrictive states

The control statement in this example specifies that the REPAIR utility is to reset the following restrictive states for the indicated objects:

- For all indexes on table spaces DBNI1601.TSNI1601 and DBNI1601.TSNI1602, reset RBDP, PSRBDP, or RBDP* status.
- For partition 1 of table space DBNI1601.TSNI1601 and partition 4 of table space DBNI1601.TSNI1602, reset ACHKP status.
- For partitions 1 and 4 of table space DBNI1601.TSNI1601, reset CHKP status.

Example: Updating version information

The control statement in this example specifies that REPAIR is to update the version information in the catalog and directory for table spaces TLKQAST1, TSKQAST2, and TPKQAST3.
Example: Repairing a table space with clones

The control statement specifies that REPAIR is to reset the auxiliary CHECK-pending (ACHKP) status of the specified table space and process only the specified objects that are table spaces that contain clone tables, indexes on clone tables, or index spaces that contain indexes on clone tables.

REPAIR
SET TABLESPACE DBKQAST1.TLKQAST1
NOAUXCHKP CLONE

Example: Checking for incorrect information in the catalog

Assume that you ran the DSN1COPY utility and want to make sure that you did not introduce any data integrity errors. Db2 automatically detects any data and catalog inconsistencies the first time that the data set is physically open after being populated by DSN1COPY. However, if you want to proactively check for these inconsistencies, you can use REPAIR. In this case, suppose that one of the affected table spaces is DBNAMET01.TSNAMET01. Issue the following REPAIR statement:

REPAIR CATALOG TABLESPACE DBNAMET01.TSNAMET01 TEST

This utility control statement specifies that REPAIR is to check for any inconsistencies between the data and catalog. If any inconsistencies are found, Db2 returns messages for them but does not correct them.

If you want REPAIR to correct the catalog when possible, issue the statement without the TEST option, as follows:

REPAIR CATALOG TABLESPACE DBNAMET01.TSNAMET01

For more information about which inconsistencies are automatically corrected and which are reported through messages, see the description of CATALOG in “Syntax and options of the REPAIR control statement” on page 670.
Chapter 30. REPORT

The REPORT utility provides information about table spaces, tables, and indexes. You can use REPORT to find the names of related table spaces, such as referentially related table spaces and LOB table spaces. You can also use REPORT to find information that is necessary for recovery.

Output

The output from REPORT with the TABLESPACESET option consists of the names of all table spaces in the table space set that you specify. It also lists all tables in the table spaces and all tables that are dependent on those tables.

The output from REPORT with the RECOVERY option consists of the following items:

- The recovery history from the SYSIBM.SYSCOPY catalog table
- Log ranges from the SYSIBM.SYSLGRNX directory table
- Volume serial numbers where archive log data sets from the BSDS exist.
- Information about any indexes on the table space that are in the informational COPY-pending (ICOPY) status (This information affects the recoverability of an index.)
- Information about any system-level backup copies that you can use for recovery if the BACKUP SYSTEM utility is used on your system

If you use system-level backup copies as the base for object-level recoveries of individual table spaces or index spaces, the REPORT output also lists the system-level backup copies. These copies are listed in the SYSCOPY ROWS AND SYSTEM-LEVEL BACKUPS section of the report.

If REPORT TABLESPACESET or REPORT RECOVERY is specified and the base objects have been cloned, information for both base and clone objects are included in the output.

In a data sharing environment, the REPORT output provides:

- The RBA of when Db2 was migrated to Db2 11
- The high and low RBA values of the migrated member
- A list of any SYSLGRNX records from before data sharing was enabled that cannot be used to recover to any point in time after data sharing was enabled
- For SYSCOPY, the member from which the image copy was deleted
- Information about system-level backup copies that are retrieved from the bootstrap data sets of each member in the data sharing group
- The status of deactivated members, and of destroyed members whose member IDs have not been reclaimed

Authorization required:

To execute this utility, you must use a privilege set that includes one of the following authorities:

- RECOVERDB privilege for the database
• DBADM or DBCTRL authority for the database. If the object on which the utility operates is in an implicitly created database, DBADM authority on the implicitly created database or DSND04 is required.
• System DBADM authority
• DATAACCESS authority
• SYSCTRL or SYSADM authority

Any of the following IDs can run the REPORT utility on any table space in DSND01 (the directory) or DSND06 (the catalog)
• An ID with DBCTRL or DBADM authority over database DSND06
• Any ID with installation SYSOPR, SYSCTRL, or SYSADM authority.

Execution phases of REPORT

The REPORT utility operates in these phases:

<table>
<thead>
<tr>
<th>Phase</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTILINIT</td>
<td>Performs initialization</td>
</tr>
<tr>
<td>REPORT</td>
<td>Collects information</td>
</tr>
<tr>
<td>UTILTERM</td>
<td>Performs cleanup</td>
</tr>
</tbody>
</table>

Related concepts:
- “Preparing for recovery by using the COPY utility” on page 170

Related tasks:
- Deleting data sharing members (Db2 Data Sharing Planning and Administration)
- Restoring deactivated data sharing members (Db2 Data Sharing Planning and Administration)

Related reference:
- “Informational COPY-pending status” on page 1042

Syntax and options of the REPORT control statement

The REPORT utility control statement, with its multiple options, defines the function that the utility job performs.

You can create a control statement with the ISPF/PDF edit function. After you create it, save it in a sequential or partitioned data set. When you create the JCL for running the job, use the SYSIN DD statement to specify the name of the data set that contains the utility control statement.
Option descriptions

RECOVERY

Indicates that recovery information for the specified table space or index is to be reported.

You can request recovery information for the local site, the recovery site, or both by using the LOCALSITE and RECOVERYSITE options. If you request
recovery information about the catalog and directory, specify the CURRENT option to avoid unnecessary mounting of archive tapes.

**Recommendation:** For image copies of partitioned table spaces that are taken with the DSNUM ALL option, run REPORT RECOVERY DSNUM ALL. If you run REPORT RECOVERY DSNUM ALL CURRENT, Db2 reports extra historical information that dates back to the last full image copy that was taken for the entire table space.

For a description of the information that REPORT RECOVERY provides, see "REPORT output" on page 710.

**TABLESPACE** `database-name.table-space-name`

For REPORT RECOVERY, specifies the table space (and, optionally, the database to which it belongs) that is being reported.

For REPORT TABLESPACESET, specifies a table space (and, optionally, the database to which it belongs) in the table space set.

`database-name`

Optionally specifies the database to which the table space belongs.

`table-space-name`

Specifies the table space.

**LIST** `listdef-name`

Specifies the name of a previously defined LISTDEF list name. The utility allows one LIST keyword for each control statement of REPORT. The list must contain only table spaces. Do not specify LIST with the TABLESPACE...table-space-name specification. The TABLESPACE keyword is required to validate the contents of the list. REPORT RECOVERY TABLESPACE is invoked once per item in the list.

**SHOWDSNS**

Specifies that the VSAM data set names for each table space or index space are to be included in the TABLESPACESET report. Data set names for base objects are shown in the section titled TABLESPACE SET REPORT. Data set names for CLONE objects are shown in the section titled CLONE TABLESPACE SET REPORT. The later report is only prepared if the base objects have been cloned.

**INDEXSPACE** `database-name.index-space-name`

Specifies the index space that is being reported.

`database-name`

Optionally specifies the database to which the index space belongs.

`index-space-name`

Specifies the index space name for the index that is being reported.

**LIST** `listdef-name`

Specifies the name of a previously defined LISTDEF list name. The utility allows one LIST keyword for each control statement of REPORT. The list must contain only index spaces. Do not specify LIST with the INDEXSPACE index-space-name specification. The INDEXSPACE keyword is required in order to validate the contents of the list. REPORT RECOVERY INDEXSPACE is invoked once for each item in the list.

**INDEX** `creator-id.index-name`

Specifies the index in the index space that is being reported.
creator-id
Optionally specifies the creator of the index.

index-name
Specifies the index name that is to be reported. Enclose the index name in quotation marks if the name contains a blank.

LIST listdef-name
Specifies the name of a previously defined LISTDEF list name. The utility allows one LIST keyword for each control statement of REPORT. The list must contain only index spaces. Do not specify LIST with the INDEX...index-name specification. The INDEX keyword is required to validate the contents of the list. REPORT RECOVERY INDEX is invoked once for each item in the list.

The partitions or partition ranges can be specified in a list.

The following REPORT keywords are optional:

INDEX NONE
Specifies that recovery information for index spaces that are associated with the specified table space is not to be reported.

INDEX ALL
Specifies that recovery information for index spaces that are associated with the specified table space is to be reported.

DSNUM
Identifies a partition or data set for which information is to be reported. Alternatively, DSNUM specifies that information is to be reported for the entire table space or index space.

ALL
Specifies that information is to be reported for the entire table space or index space.

integer
Is the number of a partition or data set for which information is to be reported. The maximum is 4096.

For a partitioned table space or partitioned index space, the integer is its physical partition number.

For a nonpartitioned table space, find the integer at the end of the data set name, as cataloged in the VSAM catalog. The data set name has the following format:

catname.DSNDBx.dbname.tsname.y0001.Annn

In this format:

catname
Is the VSAM catalog name or alias.

x Is C or D.

dbname
Is the database name.

tsname
Is the table space name.

y Is I or J.
nnn Is the data set integer.

**CURRENT**

Specifies that only the SYSCOPY entries that were written after the last recovery point of the table space are to be reported. The last recovery point is the last full image copy, LOAD REPLACE LOG YES image copy, or REORG LOG YES image copy. If you specify DSNUM ALL, the last recovery point is a full image copy that was taken for the entire table space or index space. If you specify the CURRENT option, but the last recovery point does not exist on the active log, Db2 prompts you to mount archive tapes until this point is found.

CURRENT also reports only the SYSLGRNX rows and archive log volumes that were created after the last incremental image copy entry. If no incremental image copies were created, only the SYSLGRNX rows and archive log volumes that were created after the last recovery point are reported.

If you do not specify CURRENT or if no last recovery point exists, all SYSCOPY and SYSLGRNX entries for that table space or index space are reported. The report includes entries on archive logs. If you do not specify CURRENT, the entries that were written after the last recovery point are marked with an asterisk (*) in the report.

**SUMMARY**

Specifies that only a summary of volume serial numbers is to be reported. It reports the following volume serial numbers:

- Where the archive log data sets from the BSDS exist
- Where the image copy data sets from SYSCOPY exist

If you do not specify SUMMARY, recovery information is reported, in addition to the summary of volume serial numbers.

**LOCALSITE**

Specifies that all SYSCOPY records that were copied from a local site system are to be reported.

**RECOVERYSITE**

Specifies that all SYSCOPY records that were copied from the recovery site system are to be reported.

**ARCHLOG**

Specifies which archive log data sets are to be reported.

1 Reports archive log data set 1 only.

2 Reports archive log data set 2 only.

ALL Reports both archive log data sets 1 and 2.

**TABLESPACESET**

Indicates that the names of all table spaces in the table space set and the names of all indexes on those tables are to be reported.

For more information about table space sets, see the description of the TABLESPACESET option of the QUIESCE utility.

**Related information:**

- "Syntax and options of the QUIESCE control statement" on page 416

**Related reference:**

Chapter 18, “LISTDEF,” on page 215
Data sets that REPORT uses

The REPORT utility uses a number of data sets during its operation.

The following table lists the data sets that REPORT uses. The table lists the DD name that is used to identify the data set, a description of the data set, and an indication of whether it is required. Include statements in your JCL for each required data set and any optional data sets that you want to use.

Table 103. Data sets that REPORT uses

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSIN</td>
<td>Input data set that contains the utility control statement.</td>
<td>Yes</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>Output data set for messages.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

The following object is named in the utility control statement and does not require a DD statement in the JCL:

Table space
Object that is to be reported.

Related concepts:
“Data sets that online utilities use” on page 16

Concurrency and compatibility for REPORT

The REPORT utility has certain concurrency and compatibility characteristics associated with it.

REPORT does not set a utility restrictive state on the target table space or partition.

REPORT can run concurrently on the same target object with any utility or SQL operation.

Running REPORT on the catalog and directory

REPORT RECOVERY shows the image copies for those table spaces that are not included in SYSIBM.SYSCOPY: DSNDB01.SYSUTILX, DSNDB01.DBD01, DSNDB06.SYSTSCP, and DSNDB01.SYSDBDXA.

When you run REPORT RECOVERY on one of these table spaces, specify the CURRENT option to avoid unnecessarily mounting archive tapes. If you do not specify CURRENT, Db2 searches for and reports all SYSCOPY records in the log, including those records on archive tapes. If you specify CURRENT, Db2 prompts you to mount archive tapes only if the last recovery point does not exist on the active log. You are prompted to mount tapes until the last recovery point is found.

You can use REPORT TABLESPACESET on the Db2 catalog and directory table spaces.

Termination or restart of REPORT

You can terminate and restart the REPORT utility.
You can terminate a REPORT utility job with the **TERM UTILITY** command if you have submitted the job or have SYSOPR, SYSCTRL, or SYSADM authority.

You can restart a REPORT utility job, but it starts from the beginning again.

**Related tasks:**

- "Restarting an online utility" on page 44

## REPORT output

The output from the REPORT utility depends on whether the TABLESPACESET or RECOVERY option is specified.

For the TABLESPACESET option, the output consists of the names of all table spaces in the specified table space set. For the RECOVERY option, the output includes information about the image copy data sets and archive log data set that might be required during the recovery.

### REPORT TABLESPACESET output

The output from REPORT TABLESPACESET consists of the names of all table spaces in the table space set that you specify. It also identifies all tables in the table spaces and all tables that are dependent on those tables, including LOB and XML tables, history tables, and archive tables.

#### Example of REPORT TABLESPACESET output

The statement REPORT TABLESPACESET TABLESPACE DSN8DB1A.DSN8S81D generates the output that is shown in the following figure. For the purposes of this example, an XML column was added to the sample table DSN8B10.DEPT.

```
DSNU000I 270 14:18:14.71 DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = REP94
DSNU1044I 270 14:18:14.91 DSNUGTS - PROCESSING SYSIN AS EBCDIC
DSNU050I 270 14:18:14.92 DSNUGUTC - REPORT TABLESPACESET TABLESPACE DSN8DB1A.DSN8S81D
DSNU587I 270 14:18:14.94 DSNUUPSET - REPORT TABLESPACE SET WITH TABLESPACE DSN8DB1A.DSN8S81D

TABLESPACE SET REPORT:

TABLESPACE : DSN8DB1A.DSN8S81D
  TABLE : DSN8B10.DEPT
  INDEXSPACE : DSN8DB1A.XDEPT1
  INDEX : DSN8B10.XDEPT11
  INDEXSPACE : DSN8DB1A.XDEPT2
  INDEX : DSN8B10.XDEPT22
  INDEXSPACE : DSN8DB1A.XDEPT3
  INDEX : DSN8B10.XDEPT33
  INDEXSPACE : DSN8DB1A.IROCID10D
  INDEX : DSN8B10.I_DOCIDDEPT
  DEP TABLE : DSN8B10.DEPT
              DSN8B10.EMP
              DSN8B10.PROJ

TABLESPACE : DSN8DB1A.DSN8S81E
  TABLE : DSN8B10.EMP
  INDEXSPACE : DSN8DB1A.XEMP1
  INDEX : DSN8B10.XEMP11
  INDEXSPACE : DSN8DB1A.XEMP2
  INDEX : DSN8B10.XEMP22
  DEP TABLE : DSN8B10.DEPT
              DSN8B10.EMPPROJACT
              DSN8B10.PROJ

TABLESPACE : DSN8DB1A.DSN8S81P
```
Example of REPORT TABLESPACSET output for tables spaces that are included in versioning relationships

The output from REPORT TABLESPACSET identifies versioning relationships in the system-period temporal table space or history table space. The report also includes the related auxiliary LOB and XML table spaces on both the system-period temporal table space and history table spaces.

TABLESPACE : DBSOL11.TS001L11
TABLE         : ADMF001.TBWSOL11
INDEXSPACE    : DBSOL11.IRDICIDT
INDEX         : ADMF001.I_DICIDTBWSOL11

LOB TABLESPACE SET REPORT:

TABLESPACE : DBSOL11.TS001L11
BASE TABLE   : ADMF001.TBWSOL11
COLUMN       : BLOB1
LOB TABLESPACE : DBSOL11.TLWB1L11
AUX TABLE    : ADMF001.TBAWLOBB1L11
AUX INDEXSPACE : DBSOL11.IXDLB1L11
AUX INDEX    : ADMF001.IXDLB1L11

TABLESPACE : DBSOL11.TS001L11
Example of REPORT TABLESPACESET output for tables spaces that are included in archive relationships

The following portion of output from REPORT TABLESPACESET shows related archive objects.
Related information:

Archive-enabled tables and archive tables (Introduction to Db2 for z/OS)

REPORT RECOVERY output

REPORT RECOVERY displays all information about the image copy data sets and archive log data sets that might be required during the recovery. Specifically, REPORT RECOVERY displays the following information, each in a separate section:

- Recovery information from the SYSIBM.SYSCOPY catalog table, including the history of the following utilities: QUIESCE, COPY, LOAD, REORG, and RECOVER with the TOCOPY, TOLOGPOINT, or TORBA option. For each image copy entry, the output identifies the device type and indicates whether the copy is the primary or backup copy for the local site or the recovery site.

If the SYSTEM_LEVEL_BACKUPS subsystem parameter is set to YES, your Db2 for z/OS subsystem is configured to support object-level recoveries from system-level backups. In this case, the output also includes information about any system-level backup copies that can be used to recover an individual table space or index space. This information is included at the end of the report in a section that begins with the following message:

DSNU598I - csect-name REPORT RECOVERY SYSTEM-LEVEL BACKUPS

If the DSVOLSER column of SYSIBM.SYSCOPY is blank, REPORT RECOVERY does not display volume serial numbers for image copy data sets.

- Log ranges of the table space or index space from the SYSIBM.SYSLGRNX directory.

- Information from archive log data sets ARCHLOG1, ARCHLOG2, or both, from the bootstrap data set.

If REPORT has no data to display for one or more of these sections, the corresponding sections of the report contain the following message:

DSNU588I - NO DATA TO BE REPORTED
Information about SYSIBM.SYSCOPY rows:

Several columns in the SYSIBM.SYSCOPY catalog table contain values that can be useful in interpreting the REPORT utility output. Specifically, the combination of ICTYPE, STYPE, and TTYPE column values define the event that the SYSIBM.SYSCOPY row represents. For information about the meaning of these column values, see [SYSIBM.SYSCOPY table].

Certain events can prohibit point-in-time recoveries. For information, see [Point-in-time recovery].

You can use REPORT RECOVERY output to determine the recovery status of your objects. REPORT uses the following delimiters around the ICTYPE value to indicate certain entries:

* *  REPORT uses asterisks to denote any non-image copy entries in the SYSIBM.SYSCOPY catalog table. For example, an entry that is added by the QUIESCE utility is marked with asterisks in the REPORT output.

# #  REPORT uses number signs to denote any entries in the SYSIBM.SYSCOPY catalog table that were created before any ALTER operations were materialized. For SYSIBM.SYSCOPY entries that were inserted during the materialization of the pending definition changes, REPORT uses asterisks to denote them as non-COPY entries.

< >  REPORT uses the less than symbol and greater than symbol to denote an image copy that was made before table space partitions were rebalanced.

( )  REPORT uses parentheses to denote an image copy that was made before a LOG(NO) event. For image copies of indexes, the LOG(NO) event might have occurred on its underlying table space.

Information about SYSIBM.SYSLGRNX rows and archive logs:

REPORT uses an asterisk (*) to identify SYSIBM.SYSLGRNX rows and archive log entries after the last recovery base and to indicate which logs are needed for a full recovery to the current time. (A recovery base includes an image copy or a LOAD REPLACE LOG YES or REORG LOG YES utility event.) In the REPORT output, the asterisk is displayed to the right of the entries. If you specify the CURRENT option, only the SYSIBM.SYSLGRNX rows and the archive log entries after the last recovery base are reported, and the asterisk is not included in the report.

Related reference:

- [SYSIBM.SYSCOPY table (Db2 SQL)]
- [SYSIBM.SYSLGRNX table (Db2 SQL)]
- “Syntax and options of the REPORT control statement” on page 704

Sample REPORT control statements

Use the sample control statements as models for developing your own REPORT control statements.

Example 1: Reporting recovery information for a table space

The following control statement specifies that the REPORT utility is to provide recovery information for table space DSN8D81A.DSN8S81E.
The preceding statement produces output similar to the following output:

```
REPORT RECOVERY TABLESPACE DSN8D91A.DSN8S91E SYSCOPY ROWS
TIMESTAMP = 2006-09-27-11.40.56.074739, IC TYPE = *C*, SHR LVL = , DSNUM = 0000,
START LRSN =00003697A903
DEV TYPE = LOW
IC BACK = , STYPE = L, FILE SEQ = 0000,
PIT LRSN = 000000000000
LOW DSNUM = 0000, HIGH DSNUM = 0000, OLDEST VERSION = 0000, LOGICAL PART = 0000,
LOGGED = Y, TTYPE =
JOBNAME = , AUTHID = , CPAGESF = -1.0E+00
NPAGESF = -1.0E+00
DSNAME = DSN8D91A.DSN8S91E , MEMBER NAME =
INSTANCE = 01, RELCREATED = M

REPORT RECOVERY TABLESPACE DSN8D91A.DSN8S91E SYSCOPY ROWS
TIMESTAMP = 2006-09-27-11.41.38.341008, IC TYPE = *Z*, SHR LVL = , DSNUM = 0004,
START LRSN =000036C8EA3E
DEV TYPE = LOW
IC BACK = , STYPE = , FILE SEQ = 0000,
PIT LRSN = 000000000000
LOW DSNUM = 0000, HIGH DSNUM = 0000, OLDEST VERSION = 0001, LOGICAL PART = 0004,
LOGGED = Y, TTYPE =
JOBNAME = DSNTEJ1 , AUTHID = SYSADM , CPAGESF = -1.0E+00
NPAGESF = -1.0E+00
DSNAME = DSN8D91A.DSN8S91E , MEMBER NAME =
INSTANCE = 01, RELCREATED = M

REPORT RECOVERY TABLESPACE DSN8D91A.DSN8S91E SYSCOPY ROWS
TIMESTAMP = 2006-09-27-11.41.38.341008, IC TYPE = *Z*, SHR LVL = , DSNUM = 0001,
START LRSN =000036C8EA3E
DEV TYPE = LOW
IC BACK = , STYPE = , FILE SEQ = 0000,
PIT LRSN = 000000000000
LOW DSNUM = 0000, HIGH DSNUM = 0000, OLDEST VERSION = 0001, LOGICAL PART = 0001,
LOGGED = Y, TTYPE =
JOBNAME = DSNTEJ1 , AUTHID = SYSADM , CPAGESF = -1.0E+00
NPAGESF = -1.0E+00
DSNAME = DSN8D91A.DSN8S91E , MEMBER NAME =
INSTANCE = 01, RELCREATED = M
```
Example 2: Reporting table spaces with LOB columns

The following control statement specifies that REPORT is to provide a list of all table spaces that are related to TABLESPACE DSN8D91.A.DSN891B, which contains a table with three LOB columns. The output includes a separate section titled LOB TABLESPACE SET REPORT showing a list of related LOB table spaces and their tables, indexes, and index spaces. The base table and column to which each LOB object is related is also shown.

```
REPORT TABLESPACE SET TABLESPACE DSN8D91.A.DSN891B
```

The preceding statement produces output similar to the following output:

```
DSNU0001  277 11:19:09.40 DSNUUGUC - OUTPUT START FOR UTILITY, UTILID = REPB
DSNU1044I 277 11:19:09.59 DSNUUGTC - PROCESSING SYSIN AS EBCDIC
DSNU0501  277 11:19:09.59 DSNUUGUC - REPORT TABLESPACE SET TABLESPACE DSN8D91.A.DSN891B
```
Example 3: Reporting recovery information for a partition of a partitioned table space

The following control statement specifies that REPORT is to provide recovery information for partition 4 of table space DSN8D91A.DSN8S91E. The partition number is indicated by the DSNUM option.

REPORT RECOVERY TABLESPACE DSN8D91A.DSN8S91E

The preceding statement produces output similar to the following output:

TIMESTAMP = 2006-09-27-11.41.38.341008, IC TYPE = *Z*, SHR LVL = , DSNUM = 0004,
START LRSN = 0000363CBFA3E

Chapter 30. REPORT
DEV TYPE = , IC BACK = , STYPE = , FILE SEQ = 0000,
PIT LRSN = 000000000000
LOW DSNUM = 0000, HIGH DSNUM = 0000, OLDEST VERSION = 0001, LOGICAL PART = 0004,
LOGGED = Y, TTYPE =
JOBNAME = DSNTJEJ1, AUTHID = SYSDM, COPYPAGESF = -1.00E+00
NPAGESF = -1.00E+00, CPAGESF = -1.00E+00
DSNAME = DSNB891A.DSNB891E , MEMBER NAME = ,
INSTANCE = 01, RELCREATED = M
TIMESTAMP = 2006-09-27-11.41.51.120054, IC TYPE = *Q*, SHR LVL = , DSNUM = 0000,
START LRSN = 000036E2BA9E
DEV TYPE = , IC BACK = , STYPE = W, FILE SEQ = 0000,
PIT LRSN = 000000000000
LOW DSNUM = 0000, HIGH DSNUM = 0000, OLDEST VERSION = 0001, LOGICAL PART = 0000,
LOGGED = Y, TTYPE =
JOBNAME = DSNTJEJ1, AUTHID = SYSDM, COPYPAGESF = -1.00E+00
NPAGESF = -1.00E+00, CPAGESF = -1.00E+00
DSNAME = DSNB891A.DSNB891E , MEMBER NAME = ,
INSTANCE = 01, RELCREATED = M
TIMESTAMP = 2006-09-27-11.41.53.524797, IC TYPE = *F*, SHR LVL = R, DSNUM = 0000,
START LRSN = 000036EBB3E4
DEV TYPE = 3390, IC BACK = , STYPE = , FILE SEQ = 0000,
PIT LRSN = 000000000000
LOW DSNUM = 0001, HIGH DSNUM = 0000, OLDEST VERSION = 0000, LOGICAL PART = 0000,
LOGGED = Y, TTYPE =
JOBNAME = DSNTJEJ1, AUTHID = SYSDM, COPYPAGESF = 5.70E+01
NPAGESF = 6.70E+01, CPAGESF = 5.70E+01
DSNAME = DB2V91A.SYSCOPY.DSNB891A.DSNB891E , MEMBER NAME = ,
INSTANCE = 01, RELCREATED = M
TIMESTAMP = 2006-09-27-11.41.55.631749, IC TYPE = *Q*, SHR LVL = , DSNUM = 0000,
START LRSN = 000036EABA09A
DEV TYPE = , IC BACK = , STYPE = A, FILE SEQ = 0000,
PIT LRSN = 000000000000
LOW DSNUM = 0000, HIGH DSNUM = 0000, OLDEST VERSION = 0000, LOGICAL PART = 0000,
LOGGED = Y, TTYPE =
JOBNAME = DSNTJEJ1, AUTHID = SYSDM, COPYPAGESF = -1.00E+00
NPAGESF = -1.00E+00, CPAGESF = -1.00E+00
DSNAME = DSNB891A.DSNB891E , MEMBER NAME = ,
INSTANCE = 01, RELCREATED = M
TIMESTAMP = 2006-09-27-11.42.48.167991, IC TYPE = *X*, SHR LVL = , DSNUM = 0000,
START LRSN = 0000370CA39B
DEV TYPE = , IC BACK = , STYPE = X, FILE SEQ = 0000,
PIT LRSN = 000000000000
LOW DSNUM = 0001, HIGH DSNUM = 0000, OLDEST VERSION = 0000, LOGICAL PART = 0000,
LOGGED = Y, TTYPE =
JOBNAME = DSNTJEJ1, AUTHID = SYSDM, COPYPAGESF = -1.00E+00
NPAGESF = -1.00E+00, CPAGESF = -1.00E+00
DSNAME = DSNB891A.DSNB891E , MEMBER NAME = ,
INSTANCE = 01, RELCREATED = M
TIMESTAMP = 2006-09-27-11.42.49.027488, IC TYPE = F, SHR LVL = R, DSNUM = 0000,
START LRSN = 000037113E08
DEV TYPE = 3390, IC BACK = , STYPE = , FILE SEQ = 0000,
PIT LRSN = 000000000000
LOW DSNUM = 0001, HIGH DSNUM = 0000, OLDEST VERSION = 0000, LOGICAL PART = 0000,
LOGGED = Y, TTYPE =
JOBNAME = DSNTJEJ1, AUTHID = SYSDM, COPYPAGESF = 2.00E+01
NPAGESF = 1.60E+01, CPAGESF = 1.60E+01
DSNAME = DB2V91A.DSNB891A.DSNB891E.REORGCPY , MEMBER NAME = ,
INSTANCE = 01, RELCREATED = M

DSN5861  271 18:15:27.66 DSNUPSUM - REPORT RECOVERY TABLESPACE DSN891A.DSN891E SUMMARY
DSN5888  271 18:15:27.66 DSNUPSUM - NO DATA TO BE REPORTED
DSN5833  271 18:15:27.66 DSNUPPLR - SYSLGRNX ROWS FROM REPORT RECOVERY FOR TABLESPACE DSN891A.DSN891E
Example 4: Reporting recovery information for an index

The control statement specifies that REPORT is to provide recovery information for index DSN8810.XDEPT1.

```
//REP101 EXEC DSNUPROC,SYSTEM=V91A,UID='REP101'
//SYSIN DD *
REPORT RECOVERY INDEX DSN8910.XDEPT1 /*
```

Figure 74. Example REPORT RECOVERY statement for an index

The preceding statement produces output similar to the following output:

```
DSNU000I 270 13:51:08.82 DSNU0GTC - OUTPUT START FOR UTILITY, UTILID = REP101
DSNU1044I 270 13:51:09.04 DSNU0GTC - PROCESSING SYSIN AS EBCDIC
DSNU050I 270 13:51:09.04 DSNU0GTC - REPORT RECOVERY INDEX DSN8910.XDEPT1
DSNU581I 270 13:51:09.05 DSNU0REC - REPORT RECOVERY INDEX DSN8910.XDEPT1
DSNU593I 270 13:51:09.05 DSNU0REC - REPORT RECOVERY ENVIRONMENT RECORD:
MINIMUM RBA: 000000000000
MAXIMUM RBA: FFFFFFFF0000
MIGRATING RBA: 000000000000
DSNU582I 270 13:51:09.05 DSNU0PPC - REPORT RECOVERY INDEX DSN8910.XDEPT1 SYSCOPY ROWS
TIMESTAMP = 2006-09-27-13.30.00.027880, IC TYPE = F, SHR LVL = R, DSNUM = 0000,
START LRSN = 000000000326ADE3
DEV TYPE = 3390, IC BACK = , STYPE = , FILE SEQ = 0000,
PIT LRSN = 000000318000
LOW DSNUM = 0001, HIGH DSNUM = 0001, OLDEST VERSION = 0000, LOGICAL PART = 0000,
LOGGED = Y, TTYPE = , JOBNAME = REP101, AUTHID = SYSADE,
COPYPAGESF = 0.0E+00
CPAGESF = 0.0E+00
DSNAME = DSN8910.XDEPT1.DSN8910.XDEPT1, MEMBER NAME = ,
INSTANCE = 01, RELCREATED = M
DSNU586I 270 13:51:09.05 DSNU0SUM - REPORT RECOVERY INDEX DSN8910.XDEPT1 SUMMARY
DSNU588I 270 13:51:09.05 DSNU0SUM - NO DATA TO BE REPORTED
DSNU583I 270 13:51:09.05 DSNU0PLR - SYSLGRNX ROWS FROM REPORT RECOVERY FOR INDEX DSN8910.XDEPT1
DSNU588I 270 13:51:09.05 DSNU0PLR - NO DATA TO BE REPORTED
DSNU584I 270 13:51:09.05 DSNU0PBS - REPORT RECOVERY INDEX DSN8910.XDEPT1 ARCHLOG1 BSDS VOLUMES
DSNU588I 270 13:51:09.05 DSNU0PBS - NO DATA TO BE REPORTED
DSNU586I 270 13:51:09.05 DSNU0SUM - REPORT RECOVERY INDEX DSN8910.XDEPT1 SUMMARY
DSNU588I 270 13:51:09.05 DSNU0SUM - NO DATA TO BE REPORTED
DSNU589I 270 13:51:09.05 DSNU0REC - REPORT RECOVERY INDEX DSN8910.XDEPT1 COMPLETE
```
Example 5: Reporting table space set information with XML columns

The following utility control statement specifies that REPORT is to list all objects that are related to the base table space.

REPORT TABLESPACESET TABLESPACE DBKQAN01.TPKQAN01

In this example, the base table includes two XML columns. The following output for this REPORT statement lists the objects that were implicitly created to store data for those XML columns.

Figure 75. Example output for REPORT TABLESPACESET

TABLESPACE SET REPORT:

TABLESPACE : DBKQAN01.TPKQAN01
TABLE : ADMF001.TBKQAN01
INDEXSPACE : DBKQAN01.IPKQAN11
INDEX : ADMF001.IPKQAN111
INDEXSPACE : DBKQAN01.IRDOCIDT
INDEX : ADMF001.I_DOCIDTBKQAN01
INDEXSPACE : DBKQAN01.IXKQAN12
INDEX : ADMF001.IXKQAN122

XML TABLESPACE SET REPORT:

TABLESPACE : DBKQAN01.TPKQAN01
BASE TABLE : ADMF001.TBKQAN01
COLUMN : XML1
XML TABLESPACE : DBKQAN01.XTBKQ000
XML TABLE : ADMF001.XTBKQAN01
XML NODEID INDEXSPACE : DBKQAN01.IRNODEID
XML NODEID INDEX : ADMF001.I_NODEIDXTBKQAN01
XML INDEXSPACE : DBKQAN01.XVIXLC11
XML INDEX : ADMF001.XVIXLC11
COLUMN : XML2
XML TABLESPACE : DBKQAN01.XTBKQ000
XML TABLE : ADMF001.XTBKQAN01000
XML NODEID INDEXSPACE : DBKQAN01.IRNODEID
XML NODEID INDEX : ADMF001.I_NODEIDXTBKQAN01000
XML INDEXSPACE : DBKQAN01.XVIXLC12
XML INDEX : ADMF001.XVIXLC12

Example 6: Reporting versioning relationships for system-period temporal table spaces

The following utility control statement specifies that REPORT is to list all objects that are related to the base table space.

REPORT TABLESPACESET TABLESPACE DBSOL11.TS001L11

The following report shows the objects that are involved in versioning relationships. The report also includes related auxiliary LOB and XML table spaces on both the system-period temporal table space and the history table spaces.
TABLESPACE SET REPORT:

<table>
<thead>
<tr>
<th>TABLESPACE</th>
<th>DBSOL11.TS001L11</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE</td>
<td>ADMF001.TBWSOL11</td>
</tr>
<tr>
<td>INDEXSPACE</td>
<td>DBSOL11.IRDOCIDT</td>
</tr>
<tr>
<td>INDEX</td>
<td>ADMF001.I_DOCIDTBWSOL11</td>
</tr>
</tbody>
</table>

LOB TABLESPACE SET REPORT:

<table>
<thead>
<tr>
<th>TABLESPACE</th>
<th>DBSOL11.TS001L11</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASE TABLE</td>
<td>ADMF001.TBWSOL11</td>
</tr>
<tr>
<td>COLUMN</td>
<td>BLOB1</td>
</tr>
<tr>
<td>LOB TABLESPACE</td>
<td>DBSOL11.TLWB1L11</td>
</tr>
<tr>
<td>AUX TABLE</td>
<td>ADMF001.TBAWLOBB1L11</td>
</tr>
<tr>
<td>AUX INDEXSPACE</td>
<td>DBSOL11.IXDLB1L1</td>
</tr>
<tr>
<td>AUX INDEX</td>
<td>ADMF001.IXDLB1L1</td>
</tr>
</tbody>
</table>

XML TABLESPACE SET REPORT:

<table>
<thead>
<tr>
<th>TABLESPACE</th>
<th>DBSOL11.TS001L11</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASE TABLE</td>
<td>ADMF001.TBWSOL11</td>
</tr>
<tr>
<td>COLUMN</td>
<td>XML1</td>
</tr>
<tr>
<td>XML TABLESPACE</td>
<td>DBSOL11.XTBW0000</td>
</tr>
<tr>
<td>XML TABLE</td>
<td>ADMF001.XTBWSOL11</td>
</tr>
<tr>
<td>XML NODEID INDEXSPACE</td>
<td>DBSOL11.IRNODEID</td>
</tr>
<tr>
<td>XML NODEID INDEX</td>
<td>ADMF001.I_NODEIDXTBWSOL11</td>
</tr>
<tr>
<td>XML INDEXSPACE</td>
<td>DBSOL11.IXW11SOL</td>
</tr>
<tr>
<td>XML INDEX</td>
<td>ADMF001.IXW11SOL11</td>
</tr>
<tr>
<td>XML INDEXSPACE</td>
<td>DBSOL11.IXW12SOL</td>
</tr>
<tr>
<td>XML INDEX</td>
<td>ADMF001.IXW12SOL11</td>
</tr>
<tr>
<td>XML INDEXSPACE</td>
<td>DBSOL11.IXW13SOL</td>
</tr>
<tr>
<td>XML INDEX</td>
<td>ADMF001.IXW13SOL11</td>
</tr>
<tr>
<td>XML INDEXSPACE</td>
<td>DBSOL11.IXW14SOL</td>
</tr>
<tr>
<td>XML INDEX</td>
<td>ADMF001.IXW14SOL11</td>
</tr>
</tbody>
</table>

HISTORY TABLESPACE SET REPORT:

| BASE TABLE       | ADMF001.TBWSOL11 |
| HISTORY TABLESPACE | DBSOL11.HTS001L11 |
| HISTORY TABLE    | ADMF001.HTBWSOL11 |
| HISTORY INDEXSPACE | DBSOL11.HIRDOCIDT |
| HISTORY INDEX    | ADMF001.HI_DOCIDTBWSOL11 |

HISTORY LOB TABLESPACE SET REPORT:

<table>
<thead>
<tr>
<th>HISTORY TABLESPACE</th>
<th>DBSOL11.HTS001L11</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASE TABLE</td>
<td>ADMF001.TBWSOL11</td>
</tr>
<tr>
<td>COLUMN</td>
<td>BLOB1</td>
</tr>
<tr>
<td>HISTORY LOB TABLESPACE</td>
<td>DBSOL11.HTLWB1L11</td>
</tr>
<tr>
<td>AUX TABLE</td>
<td>ADMF001.HTBWLOBB1L11</td>
</tr>
<tr>
<td>AUX INDEXSPACE</td>
<td>DBSOL11.HIXDLB1L1</td>
</tr>
<tr>
<td>AUX INDEX</td>
<td>ADMF001.HIXDLB1L1</td>
</tr>
</tbody>
</table>

HISTORY XML TABLESPACE SET REPORT:

<table>
<thead>
<tr>
<th>HISTORY TABLESPACE</th>
<th>DBSOL11.HTS001L11</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASE TABLE</td>
<td>ADMF001.TBWSOL11</td>
</tr>
<tr>
<td>COLUMN</td>
<td>XML1</td>
</tr>
<tr>
<td>HISTORY XML TABLESPACE</td>
<td>DBSOL11.HXTBW0000</td>
</tr>
<tr>
<td>XML TABLE</td>
<td>ADMF001.HXTBSOL11</td>
</tr>
<tr>
<td>XML NODEID INDEXSPACE</td>
<td>DBSOL11.HRNODEID</td>
</tr>
<tr>
<td>XML NODEID INDEX</td>
<td>ADMF001.HI_NODEIDXTBWSOL11</td>
</tr>
<tr>
<td>XML INDEXSPACE</td>
<td>DBSOL11.HIXW11SOL</td>
</tr>
<tr>
<td>XML INDEX</td>
<td>ADMF001.HIXW11SOL11</td>
</tr>
<tr>
<td>XML INDEXSPACE</td>
<td>DBSOL11.HIXW12SOL</td>
</tr>
</tbody>
</table>
Example 7: Reporting related archive tables

This example uses the following tables:

**TB_WAREHOUSE_SEG**
An application-period temporal table.

**TB_DISTRICT_SEG**
A regular table that has a referential constraint that is dependent on table TB_WAREHOUSE_SEG.

**TB_STOCK_PART**
An archive-enabled table that has a referential constraint that is dependent on table TB_WAREHOUSE_SEG.

**TB_ORDER_PBR**
A system-period temporal table that has a referential constraint that is dependent on table TB_DISTRICT_SEG.

**TB_ORDERLINE_PBG**
An archive-enabled table that has referential constraints that are dependent on tables TB_STOCK_PART and TB_ORDER_PBR.

These tables were created by the following SQL:

```sql
CREATE TABLE SC516801.TB_WAREHOUSE_SEG
(WAREHOUSE_CREATE_XML1 XML,
WAREHOUSE_ID CHAR(10) NOT NULL WITH DEFAULT,
WAREHOUSE_NAME CHAR(10) NOT NULL WITH DEFAULT,
WAREHOUSE_STREET_1 VARCHAR(40) NOT NULL WITH DEFAULT,
WAREHOUSE_STREET_2 VARCHAR(40) NOT NULL WITH DEFAULT,
WAREHOUSE_CITY VARCHAR(20) NOT NULL WITH DEFAULT,
WAREHOUSE_STATE CHAR(2) NOT NULL WITH DEFAULT,
WAREHOUSE_ZIP CHAR(9) NOT NULL WITH DEFAULT,
WAREHOUSE_TAX DECIMAL(5,4) NOT NULL WITH DEFAULT,
WAREHOUSE_YTD US_DOLLAR NOT NULL WITH DEFAULT,
WAREHOUSE_CREATE_BIGINT1 BIGINT NOT NULL WITH DEFAULT,
WAREHOUSE_CREATE_BINARY1 BINARY(101) NOT NULL WITH DEFAULT,
WAREHOUSE_CREATE_VARBINARY1 VARBINARY(500) NOT NULL WITH DEFAULT,
WAREHOUSE_CREATE_DECFLOAT1 DECFLOAT(34) NOT NULL WITH DEFAULT,
BUS_START DATE NOT NULL,
BUS_END DATE NOT NULL,
PERIOD BUSINESS_TIME(BUS_START,BUS_END),
PRIMARY KEY(WAREHOUSE_ID)
)"
```
CREATE TABLE SC516801.TB_DISTRIBUTION SEG
(DISTRICT_ID CHAR(2) NOT NULL WITH DEFAULT,
DISTRICT_WAREHOUSE_ID CHAR(10) NOT NULL WITH DEFAULT,
DISTRICT_NAME CHAR(20) NOT NULL WITH DEFAULT,
DISTRICT_CITY CHAR(20) NOT NULL WITH DEFAULT,
DISTRICT_STATE CHAR(2) NOT NULL WITH DEFAULT,
DISTRICT_ZIP CHAR(9) NOT NULL WITH DEFAULT,
DISTRICT_WAREHOUSE_ID CHAR(24) NOT NULL WITH DEFAULT,
DISTRICT_CREATE_CLOB1 CLOB(2K) NOT NULL WITH DEFAULT,
DISTRICT_CREATE_DECFLOAT1 DECFLOAT(16) NOT NULL WITH DEFAULT,
DISTRICT_CREATE_BIGINT1 BIGINT NOT NULL WITH DEFAULT,
DISTRICT_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
DISTRICT_CREATE_CHAR(40) NOT NULL WITH DEFAULT,
DISTRICT_CREATE_VARCHAR(40) NOT NULL WITH DEFAULT,
DISTRICT_CREATE_BLOB(2K) NOT NULL WITH DEFAULT,
DISTRICT_CREATE_BLOB(1000) NOT NULL WITH DEFAULT,
PRIMARY KEY (DISTRICT_WAREHOUSE_ID, DISTRICT_ID),
FOREIGN KEY (DISTRICT_WAREHOUSE_ID)
REFERENCES SC516801.TB_WAREHOUSE SEG(WAREHOUSE_ID)
ON DELETE CASCADE)
IN DBS516801.TS516801;
COMMIT;

CREATE TABLE SC516801.TB_STOCK_PART
(STOCK_ITEM_ID CHAR(6) NOT NULL WITH DEFAULT,
STOCK_WAREHOUSE_ID CHAR(10) NOT NULL WITH DEFAULT,
STOCK_QUANTITY INTEGER NOT NULL WITH DEFAULT,
STOCK_YTD INTEGER NOT NULL WITH DEFAULT,
STOCK_ORDER_CNT SMALLINT NOT NULL WITH DEFAULT,
STOCK_REMOTE_CNT SMALLINT NOT NULL WITH DEFAULT,
STOCK_WAREHOUSE_ID CHAR(24) NOT NULL WITH DEFAULT,
STOCK_WAREHOUSE_ID CHAR(24) NOT NULL WITH DEFAULT,
STOCK_WAREHOUSE_ID CHAR(24) NOT NULL WITH DEFAULT,
STOCK_WAREHOUSE_ID CHAR(24) NOT NULL WITH DEFAULT,
STOCK_WAREHOUSE_ID CHAR(24) NOT NULL WITH DEFAULT,
STOCK_WAREHOUSE_ID CHAR(24) NOT NULL WITH DEFAULT,
STOCK_DATA CHAR(50),
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
STOCK_CREATE_BLOB1 BLOB NOT NULL WITH DEFAULT,
| STOCK_CREATE_DECFLOAT1 DECFLOAT(34) NOT NULL WITH DEFAULT, |
| STOCK_CREATE_RTC1 TIMESTAMP NOT NULL |
| GENERATED BY DEFAULT FOR EACH ROW ON UPDATE |
| AS ROW CHANGE TIMESTAMP IMPLICITLY HIDDEN, |
| SYS_START TIMESTAMP(12) NOT NULL WITH DEFAULT |
| SYS_END TIMESTAMP(12) NOT NULL WITH DEFAULT |
| TRANS_ID TIMESTAMP(12) NOT NULL WITH DEFAULT |
| PRIMARY KEY (STOCK_WAREHOUSE_ID, STOCK_ITEM_ID), |
| FOREIGN KEY (STOCK_WAREHOUSE_ID) REFERENCES SC516801.TB_WAREHOUSE_SEG(WAREHOUSE_ID) |
| ON DELETE CASCADE |
| ) |
| PARTITION BY (STOCK_WAREHOUSE_ID, STOCK_ITEM_ID) |
| (PARTITION 1 ENDING ('0000000002', '999999'), |
| PARTITION 2 ENDING ('0000000004', '999999'), |
| PARTITION 3 ENDING ('0000000006', '999999'), |
| PARTITION 4 ENDING ('0000000008', '999999'), |
| PARTITION 5 ENDING ('0000000011', '999999')) |
| IN DB516803.TP516805; |
| COMMIT; |
| ... |
| ALTER TABLE TB_STOCK_PART |
| ENABLE ARCHIVE USE TB_STOCK_PBR_ARCH; |
| ... |

CREATE TABLE SC516801.TB_ORDER_PBR |
| (ORDER_ID INTEGER NOT NULL WITH DEFAULT, |
| ORDER_DISTRICT_ID CHAR(2) NOT NULL WITH DEFAULT, |
| ORDER_WAREHOUSE_ID CHAR(10) NOT NULL WITH DEFAULT, |
| ORDER_CUSTOMER_ID INTEGER GENERATED BY DEFAULT |
| AS IDENTITY (START WITH 1, INCREMENT BY 1) UNIQUE, |
| ORDER_CARRIER_ID CHAR(2), |
| ORDER_ORDERLINE_COUNT SMALLINT NOT NULL WITH DEFAULT, |
| ORDER_ALL_LOCAL SMALLINT NOT NULL WITH DEFAULT, |
| ORDER_ENTRY_DATE TIMESTAMP NOT NULL WITH DEFAULT |
| '2008-02-01-01.59.59.000000', |
| ORDER_ESTIMATE_DATE DATE DEFAULT '2008-01-01', |
| ORDER_ESTIMATE_TIME TIME DEFAULT '01.59.59', |
| ORDER_SHIP_DATE DATE NOT NULL DEFAULT '2008-01-01', |
| ORDER_SHIP_TIME TIME NOT NULL DEFAULT '01.59.59', |
| ORDER_CREATE_VARBINARY VARBINARY(500) NOT NULL WITH DEFAULT, |
| ORDER_CREATE_BIGINT BIGINT NOT NULL WITH DEFAULT, |
| ORDER_CREATE_BINARY BINARY(80) NOT NULL WITH DEFAULT, |
| ORDER_CREATE_DECFLOAT DECFLOAT(34) NOT NULL WITH DEFAULT |
| IMPLICITLY HIDDEN, |
| ORDER_CREATE_CLOB CLOB(2K) NOT NULL WITH DEFAULT, |
| ORDER_CREATE_BLOB BLOB(2K) NOT NULL WITH DEFAULT, |
| ORDER_CREATE_XML XML, |
| ORDER_CREATE_RTC1 TIMESTAMP NOT NULL |
| GENERATED ALWAYS FOR EACH ROW ON UPDATE |
| AS ROW CHANGE TIMESTAMP |
| IMPLICITLY HIDDEN, |
| SYS_START TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN, |
| SYS_END TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END |
| , |
| TRANS_ID TIMESTAMP(12) GENERATED ALWAYS AS TRANSACTION START ID, |
| PERIOD SYSTEM_TIME(SYS_START, SYS_END) |
| PRIMARY KEY (ORDER_WAREHOUSE_ID,
```
ORDER_DISTRICT_ID,  
ORDER_ID),  
FOREIGN KEY (ORDER_WAREHOUSE_ID,  
ORDER_DISTRICT_ID)  
REFERENCES SC516801.TB_DISTRICT_SEG(DISTRICT_WAREHOUSE_ID,  
DISTRICT_ID)  
ON DELETE CASCADE)  
PARTITION BY (ORDER_ID ASC)  
(PARTITION 1 ENDING AT (03000) INCLUSIVE,  
PARTITION 2 ENDING (06000) INCLUSIVE,  
PARTITION 3 ENDING (2147483647) INCLUSIVE)  
IN DB516803.TU516803;
...
ALTER TABLE SC516801.TB_ORDER_PBR  
ADD VERSIONING USE HISTORY TABLE SC516801.TB_ORDER_SEG_HIST;
...
*---------------------------------------------------
* Table: SC516801.TB_ORDERLINE_PBG bi-temporal table
* Unique Index: SC516801.IU01_ORDERLINE_PBG
* Index: SC516801.IX01_ORDERLINE_PBG
* Index on Exp: SC516801.IX02_ORDERLINE_PBG
* View: SC516801.VW_ORDERLINE_PBG
*---------------------------------------------------------------------
CREATE TABLE SC516801.TB_ORDERLINE_PBG
  (ORDERLINE_CREATE_XML1 XML,  
ORDERLINE_CREATE_LOB CLOB(2K),  
ORDERLINE_ORDER_ID INTEGER NOT NULL WITH DEFAULT,  
ORDERLINE_DISTRICT_ID CHAR(2) NOT NULL WITH DEFAULT,  
ORDERLINE_WAREHOUSE_ID CHAR(10) NOT NULL WITH DEFAULT,  
ORDERLINE_CATEGORY SMALLINT NOT NULL WITH DEFAULT,  
ORDERLINE_BARCODE1 DOUBLE,  
ORDERLINE_BARCODE2 REAL,  
ORDERLINE_BARCODE3 DOUBLE NOT NULL WITH DEFAULT,  
ORDERLINE_BARCODE4 REAL NOT NULL WITH DEFAULT,  
ORDERLINE_HASH_ENTRY1 CHAR(50) FOR BIT DATA NOT NULL WITH DEFAULT 'X'C1',  
ORDERLINE_HASH_ENTRY2 CHAR(50) FOR BIT DATA,  
ORDERLINE_UPC01 VARCHAR(100) FOR BIT DATA,  
ORDERLINE_UPC02 VARCHAR(100) FOR BIT DATA NOT NULL DEFAULT '<>',  
ORDERLINE_REMARKS VARCHAR(255) DEFAULT NULL,  
ORDERLINE_ITEM_ID CHAR(6) NOT NULL WITH DEFAULT,  
ORDERLINE_SUPPLY_WAREHOUSE_ID CHAR(10) NOT NULL WITH DEFAULT,  
ORDERLINE_DELIVERY_DATE TIMESTAMP,  
ORDERLINE_QUANTITY INTEGER NOT NULL WITH DEFAULT,  
ORDERLINE_AMOUNT US_DOLLAR NOT NULL WITH DEFAULT,  
ORDERLINE_DISTRICT_INFO CHAR(24) NOT NULL WITH DEFAULT,  
ORDERLINE_CREATE_DECFLAT1 DECFLAT(16) NOT NULL WITH DEFAULT,  
ORDERLINE_CREATE_VARBINARY1 VARBINARY(500) NOT NULL WITH DEFAULT,  
ORDERLINE_CREATE_BIGINT1 BIGINT NOT NULL WITH DEFAULT,  
ORDERLINE_CREATE_BINARY1 BINARY(30) NOT NULL WITH DEFAULT,  
SYS_START_TIMESTAMP(12) NOT NULL WITH DEFAULT,  
SYS_END_TIMESTAMP(12) NOT NULL WITH DEFAULT,  
TRANS_ID TIMESTAMP(12),  
BUS_START TIMESTAMP(6) NOT NULL,  
BUS_END TIMESTAMP(6) NOT NULL,  
CONSTRAINT PK#TB_ORDERLINE_DISTRICT_ID#ORDER_ID#CATEGORY  
PRIMARY KEY (ORDERLINE_WAREHOUSE_ID,  
ORDERLINE_DISTRICT_ID,  
ORDERLINE_ORDER_ID,  
ORDERLINE_CATEGORY),  
CONSTRAINT FK#TB_ORDER#WAREHOUSE_ID#DISTRICT_ID#ORDER_ID#CASCADE  
FOREIGN KEY (ORDERLINE_WAREHOUSE_ID,  
ORDERLINE_ORDER_ID)  
REFERENCES SC516801.TB_ORDER_PBR  
(ORDER_WAREHOUSE_ID,
ORDER_DISTRICT_ID,
ORDER_ID)
ON DELETE CASCADE,
CONSTRAINT FK#TB_STOCK_WAREHOUSE_ID#STOCK_ITEM_ID#CASCADE
FOREIGN KEY (ORDERLINE_SUPPLY_WAREHOUSE_ID,
ORDERLINE_ITEM_ID)
REFERENCES SC516801.TB_STOCK_PART
(STOCK_WAREHOUSE_ID,
STOCK_ITEM_ID)
ON DELETE CASCADE)
IN DB516807.TG516807
APPEND YES;

ALTER TABLE SC516801.TB_ORDERLINE_PBG
ENABLE ARCHIVE USE SC516801.TB_ORDERLINE_PBR_ARCH;

The following utility control statement specifies that REPORT is to list all objects
that are related to table space TS516801, which contains tables TB_DISTRICT_SEG
and TB_WAREHOUSE_SEG.
REPORT TABLESPACESET TABLESPACE DB516801.TS516801

The resulting output lists referentially related objects, related LOB and XML tables,
and related history and archive tables:

TABLESPACE SET REPORT:

TABLESPACE  : DB516801.TS516801
TABLE        : SC516801.TB_DISTRICT_SEG
INDEXSPACE   : DB516801.IU01RDIS
INDEX        : SC516801.IU01_DISTRICT_SEG
INDEXSPACE   : DB516801.IX01RDIS
INDEX        : SC516801.IX01_DISTRICT_SEG
INDEXSPACE   : DB516801.IX02RDIS
INDEX        : SC516801.IX02_DISTRICT_SEG
DEP TABLE    : SC516801.TB_ORDER_PBR

TABLE        : SC516801.TB_WAREHOUSE_SEG
INDEXSPACE   : DB516801.IU01RWAR
INDEX        : SC516801.IU01_WAREHOUSE_SEG
INDEXSPACE   : DB516801.IX01RWAR
INDEX        : SC516801.IX01_WAREHOUSE_SEG
INDEXSPACE   : DB516801.IX02RWAR
INDEX        : SC516801.IX02_WAREHOUSE_SEG
INDEXSPACE   : DB516801.IX03RWAR
INDEX        : SC516801.IX03_WAREHOUSE_SEG
INDEXSPACE   : DB516801.IX04RWAR
INDEX        : SC516801.IX04_WAREHOUSE_SEG
DEP TABLE    : SC516801.TB_DISTRICT_SEG
              : SC516801.TB_STOCK_PART

TABLESPACE  : DB516803.TP516805
TABLE        : SC516801.TB_STOCK_PART
INDEXSPACE   : DB516803.IU01RSTO
INDEX        : SC516801.IX01_STOCK_PART
INDEXSPACE   : DB516803.IU01RSTO
INDEX        : SC516801.IU01_STOCK_PART
INDEXSPACE   : DB516803.IX02RSTO
INDEX        : SC516801.IX02_STOCK_PART
DEP TABLE    : SC516801.TB_ORDERLINE_PBG

TABLESPACE  : DB516803.TU516803
TABLE        : SC516801.TB_ORDER_PBR
INDEXSPACE   : DB516803.IU01RORD
INDEX        : SC516801.IU01_ORDER_PBR
INDEXSPACE  : DB516803.IROCDIT
INDEX  : SC516801.I_DOCIDTB_ORDER_PBR
INDEXSPACE  : DB516803.IO2RORD
INDEX  : SC516801.IO02_ORDER_PBR
INDEXSPACE  : DB516803.IO3RORD
INDEX  : SC516801.IO03_ORDER_PBR
INDEXSPACE  : DB516803.IO4RORD
INDEX  : SC516801.IO04_ORDER_PBR
INDEXSPACE  : DB516803.IX01RORD
INDEX  : SC516801.IX01_ORDER_PBR
INDEXSPACE  : DB516803.IX02RORD
INDEX  : SC516801.IX02_ORDER_PBR
INDEXSPACE  : DB516803.IX03RORD
INDEX  : SC516801.IX03_ORDER_PBR
INDEXSPACE  : DB516803.IX04RORD
INDEX  : SC516801.IX04_ORDER_PBR

DEP TABLE  : SC516801.TB_ORDERLINE_PBG

TABLESPACE  : DB516807.TG516807
TABLE  : SC516801.TB_ORDERLINE_PBG
INDEXSPACE  : DB516807.IU01RORD
INDEX  : SC516801.IU01_ORDERLINE_PBG
INDEXSPACE  : DB516807.IU02RORD
INDEX  : SC516801.IU02_ORDERLINE_PBG
INDEXSPACE  : DB516807.IU03RORD
INDEX  : SC516801.IU03_ORDERLINE_PBG
INDEXSPACE  : DB516807.IU04RORD
INDEX  : SC516801.IU04_ORDERLINE_PBG
INDEXSPACE  : DB516807.IU05RORD
INDEX  : SC516801.IU05_ORDERLINE_PBG

LOB TABLESPACE SET REPORT:

TABLESPACE  : DB516801.TS516801
BASE TABLE  : SC516801.TB_DISTRICT_SEG
COLUMN  : DISTRICT_CREATE_CLOBI
LOB TABLESPACE  : DB516801.TA516801
AUX TABLE  : SC516801.TX01_CLOB1_DISTRICT_SEG
AUX INDEXSPACE  : DB516801.IARCLOB1
AUX INDEX  : SC516801.IA_CLOB1_DISTRICT_SEG

TABLESPACE  : DB516803.TU516803
BASE TABLE  : SC516801.TB_ORDER_PBR
PART: 0001 COLUMB  : ORDER_CREATE_CLOBI
LOB TABLESPACE  : DB516803.TA516831
AUX TABLE  : SC516801.TX31_CLOB_ORDER_PBR
AUX INDEXSPACE  : DB516803.IA31RCLO
AUX INDEX  : SC516801.IA31_CLOB_ORDER_PBR
PART: 0002 COLUMB  : ORDER_CREATE_CLOBI
LOB TABLESPACE  : DB516803.TA516832
AUX TABLE  : SC516801.TX32_CLOB_ORDER_PBR
AUX INDEXSPACE  : DB516803.IA32RCLO
AUX INDEX  : SC516801.IA32_CLOB_ORDER_PBR
PART: 0003 COLUMB  : ORDER_CREATE_CLOBI
LOB TABLESPACE  : DB516803.TA516833
AUX TABLE  : SC516801.TX33_CLOB_ORDER_PBR
AUX INDEXSPACE  : DB516803.IA33RCLO
AUX INDEX  : SC516801.IA33_CLOB_ORDER_PBR
PART: 0001 COLUMB  : ORDER_CREATE_BLOBI
LOB TABLESPACE  : DB516803.TA516834
AUX TABLE  : SC516801.TX31_BLOB_ORDER_PBR
AUX INDEXSPACE  : DB516803.IA31RBLO
AUX INDEX  : SC516801.IA31_BLOB_ORDER_PBR
PART: 0002 COLUMB  : ORDER_CREATE_BLOBI
LOB TABLESPACE  : DB516803.TA516835
AUX TABLE  : SC516801.TX32_BLOB_ORDER_PBR
AUX INDEXSPACE  : DB516803.IA32RBLO
AUX INDEX  : SC516801.IA32_BLOB_ORDER_PBR
PART: 0003  COLUMN : ORDER_CREATE_BLOB1
LOB TABLESPACE : DB516803.TA516836
AUX TABLE : SC516801.TX33_BLOB_ORDER_PBR
AUX INDEXSPACE : DB516803.IA33_BLOB_ORDER_PBR
AUX INDEX : SC516801.IA33_BLOB_ORDER_PBR

TABLESPACE : DB516807.TG516807

BASE TABLE : SC516801.TB_ORDERLINE_PBG

PART: 0001  COLUMN : ORDERLINE_CREATE_LOB
LOB TABLESPACE : DB516807.TA516871
AUX TABLE : SC516801.TX71_CLOB1_ORDERLINE_PBG
AUX INDEXSPACE : DB516807.IA71RCLO
AUX INDEX : SC516801.IA71_CLOB1_ORDERLINE_PBG

XML TABLESPACE SET REPORT:

TABLESPACE : DB516801.TS516801
BASE TABLE : SC516801.TB_WAREHOUSE_SEG
COLUMN : WAREHOUSE_CREATE_XML1
XML TABLESPACE : DB516801.XT80000
XML TABLE : SC516801.XTB_WAREHOUSE_SEG
XML NODEID INDEXSPACE: DB516801.IRNODEID
XML NODEID INDEX : SC516801.I_NODEIDXTB_WAREHOUSE_SEG

TABLESPACE : DB516803.TG516803
BASE TABLE : SC516801.TB_ORDER_PBR
COLUMN : ORDER_CREATE_XML
XML TABLESPACE : DB516803.XT80000
XML TABLE : SC516801.XTB_ORDER_PBR
XML NODEID INDEXSPACE: DB516803.IRNODEID
XML NODEID INDEX : SC516801.I_NODEIDXTB_ORDER_PBR

TABLESPACE : DB516807.TG516807
BASE TABLE : SC516801.TB_ORDERLINE_PBG
COLUMN : ORDERLINE_CREATE_XML1
XML TABLESPACE : DB516807.XT80000
XML TABLE : SC516801.XTB_ORDERLINE_PBG
XML NODEID INDEXSPACE: DB516807.IRNODEID
XML NODEID INDEX : SC516801.I_NODEIDXTB_ORDERLINE_PBG

HISTORY TABLESPACE SET REPORT:

TABLESPACE : DB516804.TS516804
HISTORY TABLE : SC516801.TB_ORDER_SEG_HIST
TEMPORAL TABLE : SC516801.TB_ORDER_PBR
INDEXSPACE : DB516804.IU01RORD
INDEX : SC516801.IU01_ORDER_SEG_HIST
INDEXSPACE : DB516804.IRDOCID
INDEX : SC516801.I_DOCIDTB_ORDER_SEG_HIST
INDEXSPACE : DB516804.IU02RORD
INDEX : SC516801.IU02_ORDER_SEG_HIST
INDEXSPACE : DB516804.IU03RORD
INDEX : SC516801.IU03_ORDER_SEG_HIST
INDEXSPACE : DB516804.IU04RORD
INDEX : SC516801.IU04_ORDER_SEG_HIST
INDEXSPACE : DB516804.IX01RORD
INDEX : SC516801.IX01_ORDER_SEG_HIST
INDEXSPACE : DB516804.IX02RORD
INDEX : SC516801.IX02_ORDER_SEG_HIST
INDEXSPACE : DB516804.IU05RORD
INDEX : SC516801.IU05_ORDER_SEG_HIST

HISTORY LOB TABLESPACE SET REPORT:

TABLESPACE : DB516804.TS516804

BASE TABLE : SC516801.TB_ORDER_SEG_HIST
COLUMN : ORDER_CREATE_CLOB1
LOB TABLESPACE : DB516804.TA516841
AUX TABLE : SC516801.TX41_CLOB_ORDER_SEG_HIST
AUX INDEXSPACE : DB516804.IA41RCLO
AUX INDEX : SC516801.IA41_CLOB_ORDER_SEG_HIST
COLUMN : ORDER_CREATE_BLOB1
LOB TABLESPACE : DB516804.TA516842
AUX TABLE : SC516801.TX42_BLOB_ORDER_SEG_HIST
AUX INDEXSPACE : DB516804.IA42RBLO
AUX INDEX : SC516801.IA42_BLOB_ORDER_SEG_HIST

HISTORY XML TABLESPACE SET REPORT:

TABLESPACE : DB516804.TS516804

BASE TABLE : SC516801.TB_ORDER_SEG_HIST
COLUMN : ORDER_CREATE_XML
XML TABLESPACE : DB516804.XTBRO000
XML TABLE : SC516801.XTB_ORDER_SEG_HIST
XML NODEID INDEXSPACE : DB516804.IRNODEID
XML NODEID INDEX : SC516801.I_NODEIDXTB_ORDER_SEG_HIST

ARCHIVE TABLESPACE SET REPORT:

TABLESPACE : DB516803.TU516806
ARCHIVE TABLE : SC516801.TB_STOCK_PBR_ARCH
AR_ENABLED TABLE : SC516801.TB_STOCK_PART
INDEXSPACE : DB516803.IX01RARC
INDEX : SC516801.IX01_ARCH_STOCK_PBR
INDEXSPACE : DB516803.IU01RARC
INDEX : SC516801.IU01_ARCH_STOCK_PBR
INDEXSPACE : DB516803.IX02RARC
INDEX : SC516801.IX02_ARCH_STOCK_PBR

TABLESPACE : DB516807.TU516808
ARCHIVE TABLE : SC516801.TB_ORDERLINE_PBR_ARCH
AR_ENABLED TABLE : SC516801.TB_ORDERLINE_PBG
INDEXSPACE : DB516807.IU01RARC
INDEX : SC516801.IU01_ARCH_ORDERLINE_PBR
INDEXSPACE : DB516807.IRO0TOL
INDEX : SC516801.I_DOCIDTB_ORDERLINE_PBR_A
INDEXSPACE : DB516807.IX01RARC
INDEX : SC516801.IX01_ARCH_ORDERLINE_PBR
INDEXSPACE : DB516807.IX02RARC
INDEX : SC516801.IX02_ARCH_ORDERLINE_PBR

ARCHIVE LOB TABLESPACE SET REPORT:

TABLESPACE : DB516807.TU516808

BASE TABLE : SC516801.TB_ORDERLINE_PBR_ARCH
PART: 0001 COLUMN : ORDERLINE_CREATE_LOB
LOB TABLESPACE : DB516807.TA516881
AUX TABLE : SC516801.TX81_CLOB1_ORDERLINE_PBR_ARCH
AUX INDEXSPACE : DB516807.IA81RCLO
| AUX INDEX       | SC516801.IA81_CLOB1_ORDERLINE_PBR_ARCH |
| PART: 0002     | COLUMN: ORDERLINE_CREATE_LOB          |
| LOB TABLESPACE | DB516807.TA516882                     |
| AUX TABLE      | SC516801.TX82_CLOB1_ORDERLINE_PBR_ARCH |
| AUX INDEXSPACE | DB516807.IA82RCLO                     |
| AUX INDEX      | SC516801.IA82_CLOB1_ORDERLINE_PBR_ARCH |
| PART: 0003     | COLUMN: ORDERLINE_CREATE_LOB          |
| LOB TABLESPACE | DB516807.TA516883                     |
| AUX TABLE      | SC516801.TX83_CLOB1_ORDERLINE_PBR_ARCH |
| AUX INDEXSPACE | DB516807.IA83RCLO                     |
| AUX INDEX      | SC516801.IA83_CLOB1_ORDERLINE_PBR_ARCH |

ARCHIVE XML TABLESPACE SET REPORT:

| TABLESPACE   | DB516807.TU516808 |
| BASE TABLE   | SC516801.TB_ORDERLINE_PBR_ARCH   |
| COLUMN       | ORDERLINE_CREATE_XML1           |
| XML TABLESPACE| DB516807.XTBR0001              |
| XML TABLE    | SC516801.XTB_ORDERLINE_PBR_A    |
| XML NODEID INDEXSPACE | DB516807.IRNO1LJL |
| XML NODEID INDEX | SC516801.I_NODEIDXTB_ORDERLINE_PBR_ |
Chapter 31. RESTORE SYSTEM

The RESTORE SYSTEM online utility invokes z/OS DFSMShsm to recover a Db2 subsystem or a data sharing group to a previous point in time. To perform the recovery, the utility uses data that is copied by the BACKUP SYSTEM utility.

Requirement: All data sets that are recovered with RESTORE SYSTEM must be SMS-managed.

The RESTORE SYSTEM utility can be run from any member in a data sharing group, even one that is normally quiesced when any backups are taken. Any member in the data sharing group that is active at or beyond the log truncation point must be restarted, and its logs are truncated to the SYSPITR LRSN point.

You can specify the SYSPITR LRSN point in the CRESTART control statement of the DSNJU003 (Change Log Inventory) utility. Any data sharing group member that is normally quiesced at the time the backups are taken and is not active at or beyond the log truncation point does not need to be restarted.

By default, RESTORE SYSTEM recovers the data from the database copy pool during the RESTORE phase and then applies logs to the point in time at which the existing logs were truncated during the LOGAPPLY phase. The RESTORE utility never restores logs from the log copy pool.

Restriction: RESTORE SYSTEM does not restore logs; the utility only applies the logs. If you specified BACKUP SYSTEM FULL to create copies of both the data and the logs, you can restore the logs by another method.

Output:

Output for RESTORE SYSTEM is the recovered copy of the data volume or volumes.

Authorization required

To run this utility, you must use a privilege set that includes SYSADM authority.

The user who submits the job must also have certain RACF authorities if both of the following conditions are true:

• A system-level backup on tape is the input for the RESTORE SYSTEM utility.
• The z/OS level is Version 1 Release 11 or earlier.

In this situation, the following two RACF authorities are required:

• Operations authority, as in ATTRIBUTES=OPERATIONS
• DASDVOL authority, which you can set in the following way:

  SETROPTS GENERIC(DASDVOL)
  REDEFINE DASDVOL + UACC(ALTER)
  SETROPTS CLASSACT(DASDVOL)
  SETROPTS GENERIC(DASDVOL) REFRESH

You can restrict this authority to specific user IDs.
This RACF authority is required, because the RESTORE SYSTEM utility invokes DFSMSdss when tape is the input and the z/OS level is Version 1 Release 11 or earlier. However, DFSMSdss is not invoked when you restore database copy pools from tape when the z/OS level is Version 1 Release 12 or later or from a FlashCopy on disk. In these situations, the RESTORE SYSTEM utility invokes DFSMShsm, which does not require Operations or DASDVOL authority.

**Execution phases of RESTORE SYSTEM**

The RESTORE SYSTEM utility operates in the following phases:

<table>
<thead>
<tr>
<th>Phase</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTILINIT</td>
<td>Performs initialization and setup</td>
</tr>
<tr>
<td>RESTORE</td>
<td>Locates and restores the volume copies if the LOGONLY option is not specified</td>
</tr>
<tr>
<td>LOGAPPLY</td>
<td>Applies the outstanding log changes to the database</td>
</tr>
<tr>
<td>UTILTERM</td>
<td>Performs cleanup</td>
</tr>
</tbody>
</table>

**Related concepts:**

- Point-in-time recovery with system-level backups (Db2 Administration Guide)

**Related reference:**

Chapter 8, “BACKUP SYSTEM,” on page 51

**Syntax and options of the RESTORE SYSTEM control statement**

The RESTORE SYSTEM utility control statement, with its multiple options, defines the function that the utility job performs.

Use the ISPF/PDF edit function to create a control statement and to save it in a sequential or partitioned data set. When you create the JCL for running the job, use the SYSIN DD statement to specify the name of the data set that contains the utility control statement.

When you specify RESTORE SYSTEM, you can specify only the following statements in the same step:

- DIAGNOSE
- OPTIONS PREVIEW
- OPTIONS OFF
- OPTIONS KEY
- OPTIONS EVENT WARNING

In addition, RESTORE SYSTEM must be the last statement in SYSIN.
Option descriptions

**LOGONLY**

Specifies that the database volumes have already been restored, so the RESTORE phase is skipped. Use this option when the database volumes have already been restored outside of Db2. If the subsystem is at a tracker site, you must specify the LOGONLY option.

**SWITCH VCAT**

Indicates that the integrated catalog facility (ICF) alias (VCAT) names are to be substituted with those names that are provided when the log is processed. Every VCAT encountered in the log must be specified in the SYSVALUEDDN data set. This option might be used in the process of cloning a Db2 subsystem.

**SYSVALUEDDN ('ddname')**

Specifies that the DD statement for the control statements specifying the integrated catalog facility (ICF) (VCAT) aliases used when processing log records. ddname can be up to 8 characters, and must start with an alphabetic or national character (for example: @, $, or #).

The default value is SYSVALUEDDN(SYSVALUE), where SYSVALUE identifies the primary data set.

**FROMDUMP**

Indicates that you want to dump only the database copy pool to tape during the restore.

**DUMPCLASS (dcl)**

Indicates what DFSMShsm dump class to use for the restore.

**RSA ('key-label')**

Specifies that the key-label value in the utility control statement is passed to DFSMShsm to override the key-label value that would normally be used to read dump tapes. key-label can be up to 64 characters, and must start with an alphabetic or national character (for example: @, $, or #). key-label must be enclosed in single quotation marks.

The FROMDUMP and DUMPCLASS options that you specify for the RESTORE SYSTEM utility override the RESTORE RECOVER_FROMDUMP and UTILS_DUMP_CLASS_NAME subsystem parameter values.

**TAPEUNITS**

Specifies the limit on the number of tape drives that the utility dynamically allocates during the restore of the database copy pool from dumps on tape.
The default is the option that you specified for subsystem parameter
RESTORE_TAPEUNITS. If no default is specified, then the RESTORE SYSTEM
utility tries to use all of the tape drives in your system.

The TAPEUNITS option does not apply and is ignored when the z/OS level is
Version 1 Release 12 or later.

(num-tape-units)

Specifies the maximum number of tape drives to allocate. If you specify
zero, or you do not specify a value, the utility determines the optimal
number of tape units to use. RESTORE SYSTEM TAPEUNITS has a
maximum value of 255.

Related tasks:

- Recovering from disasters by using a tracker site (Db2 Administration Guide)

---

**Before running RESTORE SYSTEM**

Certain activities might be required before you run the RESTORE SYSTEM utility, depending on your situation.

Complete the following steps prior to running RESTORE SYSTEM:

1. Stop Db2. If data sharing, stop all Db2 members in the group.
2. Run DSNJU003 (Change Log Inventory) to create a Db2 conditional restart
record with the CRESTART SYSPITR option. Specify the log truncation point
with the SYSPITR option that corresponds to the point in time to which the
system is to be recovered.

For data sharing, specify an LRSN value. For non data sharing, specify an RBA
value.

If you restored the log copy pool and the active log data sets are stripped or
the log copy pool is for a data sharing environment, you must specify the data
copy LRSN during the conditional restart in the following scenarios:

- You are cloning a Db2 system by using a system-level backup as the source.
  In this case, conditionally restart Db2 with an ENDRBA or ENDLRSN that is
equal to the data complete LRSN of the system-level backup.

- You are performing a system-level point-in-time recovery. In this case,
  conditionally restart Db2 with the log truncation point equal to or less than
  the data complete LRSN of the system-level backup. Use the data complete
  LRSN as the CRESTART ENDRBA, ENDLRSN, or SYSPITR log truncation
  point.

You can determine the data complete LRSN from the following places:

- Message DSNU1614I, which is generated when BACKUP SYSTEM completes
  successfully

- The report generated by the print log map utility (DSNJU004)

3. Start Db2. When the Db2 restart processing for the conditional restart with the
   SYSPITR option completes, Db2 enters system RECOVER-pending and access
   maintenance mode. During system RECOVER-pending mode, you can run only
   the RESTORE SYSTEM utility.

4. Ensure that all data sharing members that were active at the SYSPITR log
   truncation point (or restarted after this point) have been restarted with the
   same SYSPITR LRSN value. You can stop the other members of the data group
   (with MODE(QUIESCE)) after the SYSPITR restart.

5. Ensure that the ICF catalogs for the Db2 data are not active and are not
   allocated. The ICF catalog for the data must be on a separate volume than the
ICF catalog for the logs. The command to unallocate the catalog is `CATALOG,UNALLOCATE(catalog-name)`.
Alternatively, if you add the ICF catalog names to the database copy pool definition by altering the copy pools, the catalog is unallocated by HSM before doing the restore.

**Related information:**
[Altering copy pools (DFSMSdfp Storage Administration)]

### How to determine which system-level backups Db2 restores

The RESTORE SYSTEM utility uses the most recent system-level backup of the database copy pool that Db2 took before the SYSPITR log truncation point.

To determine whether the system level backup will be restored from disk or from tape:
- If `FROMDUMP` was not specified and the system-level backup resides on disk, Db2 uses it for the restore.
- If you specify YES in the RESTORE/RECOVER FROM DUMP field on installation panel DSNTIP6 or you specify the FROMDUMP option in the RESTORE utility statement, restore uses only the dumps on tape of the database copy pool.
- If you specify a dump class name on the DUMP CLASS NAME field on installation panel DSNTIP6 or you specify the DUMPCLASS option in the RESTORE utility statement, Db2 restores the database copy pool from the DFSMSshm dump class.
- If you do not specify a dump class name in the DUMP CLASS NAME field on installation panel DSNTIP6 or you do not specify the DUMPCLASS option in the RESTORE utility statement, RESTORE SYSTEM issues the DFSMSshm LIST COPYPOOL command and uses the first dump class listed in the output.

The RESTORE SYSTEM utility invokes DFSMS to restore the database copy pool volumes from a system-level backup on tape. If the z/OS level is Version 1 Release 11 or earlier, the utility invokes DFSMSdss. If the z/OS level is Version 1 Release 12 or later, the utility invokes DFSMSshm.

### How to determine if RESTORE SYSTEM uses parallelism when restoring from tapes
Parallelism occurs if the dumps of the volumes in the database copy pool reside on different tape volumes. The degree of parallelism is limited by:
- The TAPEUNITS option, which limits the number of tape units that the utility can allocate.
- The number of distinct tape volumes that the dump resides on.

### Determining whether the system-level backups reside on disk or tape
Restoring each volume in the database copy pool from a fast replication copy on the disk occurs virtually instantaneously. Restoring the database copy pool from dumps on tape volumes takes much longer.

To determine whether the system-level backups of the database copy pool reside on the disk or tape:
1. Run the DFSMSshm LIST COPYPOOL command with the ALLVOLS option.
2. Run the DSNJU004 utility output. For data sharing, run the DSNJU004 utility output on each member.

3. Review the output from the DFSMShsm LIST COPYPOOL command with the ALLVOLS option.

4. Review the Db2 system-level backup information in the DSNJU004 utility output.

If the system-level backup chosen as the recovery base for the database copy pool no longer resides on DASD and the FROMDUMP option has not been specified, then the RESTORE SYSTEM utility will fail. You can then specify the RESTORE SYSTEM FROMDUMP option, or specify it on installation panel DSNTIP6, to direct the utility to use the system-level backup that was dumped to tape.

### While running RESTORE SYSTEM

Specific considerations apply while RESTORE SYSTEM is running.

### Data sets that RESTORE SYSTEM uses

The following table lists the data sets that RESTORE SYSTEM uses. The table lists the DD name that is used to identify the data set, a description of the data set, and whether it is required. Include statements in your JCL for each required data set.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSIN</td>
<td>An input data set that contains the utility control statement</td>
<td>Yes</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>An output data set for messages</td>
<td>Yes</td>
</tr>
<tr>
<td>auth-id.job-name.HSM</td>
<td>A temporary data set that is automatically allocated by the utility and deleted when the utility completes</td>
<td>Yes</td>
</tr>
<tr>
<td>VCAT alias values data set</td>
<td>An input data set that contains the values of the integrated catalog facility (ICF) alias (VCAT) names to be switched while processing. Specify its DD name with the SYSVALUEDDN option of the utility control statement. The default DD name is SYSVALUE.</td>
<td>No</td>
</tr>
</tbody>
</table>

**VCAT alias values data set**

Defines a set of records which contain integrated catalog facility (ICF) catalog (VCAT) alias names.

Each record must contain a pair of (VCAT) alias names separated by only a comma. Blank characters are not allowed between each name. Each name is a valid z/OS alias of up to eight characters and composed of uppercase alphabetic, numeric, or national characters. The first (VCAT) alias name is the name used when the system level backup was created. The second (VCAT) alias name is the current name after any renaming. All aliases encountered in the log must be specified, even if the VCAT alias is the same as when the system level backup was created. Sample data follows:

VCAT1, VCAT2
VCAT5, Z1234567
DSNC000, DSNC000
To obtain the names, keep a list of previously existing to current name mappings when renaming an integrated catalog facility (ICF) catalog (VCAT) alias.

**Concurrency and compatibility for RESTORE SYSTEM**

While RESTORE SYSTEM is running, no other utilities can run.

**Using DISPLAY UTILITY with RESTORE SYSTEM**

You can use the DISPLAY UTILITY command with RESTORE SYSTEM.

To use the DISPLAY UTILITY command for RESTORE SYSTEM on a data sharing group, you must issue the command from the member on which the RESTORE SYSTEM utility is invoked.

**Termination and restart of RESTORE SYSTEM**

You can terminate and restart the RESTORE SYSTEM utility.

You cannot terminate RESTORE SYSTEM by using the TERM UTILITY command.

You can restart RESTORE SYSTEM at the beginning of a phase or at the current system checkpoint. A current system checkpoint occurs during the LOGAPPLY phase after log records are processed. By default, RESTORE SYSTEM restarts at the current system checkpoint.

When you restart RESTORE SYSTEM for a data sharing group, the member on which the restart is issued must be the same member on which the original RESTORE SYSTEM was issued.

**Related concepts:**

"Data sets that online utilities use" on page 16

**Related tasks:**

"Restarting an online utility" on page 44

**Related reference:**

DISPLAY UTILITY (Db2) (Db2 Commands)

---

**After running RESTORE SYSTEM**

The effects of the RESTORE SYSTEM utility and the activities that might be required after it runs vary, depending on your situation.

**Effects of running RESTORE SYSTEM**

**Recovering table spaces defined with NOT LOGGED attribute**

If RESTORE SYSTEM utility determines that a table space defined with the NOT LOGGED attribute was updated after the point at which the system-level copy was taken, the table space or partition is marked RECOVER-pending.

**Cases when indexes are placed in REBUILD-pending status**

When you use the RESTORE SYSTEM utility to recover indexes, an index might be left in REBUILD-pending status. In these rare cases, you must rebuild the index by running the REBUILD INDEX utility.
Indexes are left in REBUILD-pending status, if on the Db2 subsystem that is being recovered:
- Some indexes have gone through the two-pass group buffer pool recovery pending (GRECP) or logical page list (LPL) recovery earlier
- Or the indexes are still in GRECP or LPL status, and the compensation log records are written before the physical undo logs

**After running RESTORE SYSTEM**

Complete the following steps after running RESTORE SYSTEM:
1. Stop and start each Db2 subsystem or member to remove it from access maintenance mode.
2. Use the DISPLAY UTIL command to see if any utilities are running. If other utilities are running, use the TERM UTIL command to end them.
3. Use the RECOVER utility to recover all objects in RECOVER-pending (RECP) or REBUILD-pending (RBDP) status, or use the REBUILD INDEX utility to rebuild objects. If a CREATE TABLESPACE, CREATE INDEX, or data set extension has failed, you can also recover or rebuild any objects in the logical page list (LPL).

**Sample RESTORE SYSTEM control statements**

Use the sample control statements as models for developing your own RESTORE SYSTEM utility control statements.

The RESTORE SYSTEM utility uses data that is copied by the BACKUP SYSTEM utility.

**Example 1: Recovering a backup system**

The following control statement specifies that RESTORE SYSTEM is to recover a Db2 subsystem or a data sharing group to a previous point in time by restoring volume copies and applying any outstanding log changes.

```
//STEP1 EXEC DSNUPROC,TIME=1440,
  // UTPROC='',
  // SYSTEM='DSN'
  //SYSIN DD *
  RESTORE SYSTEM
/*
```

**Example 2: Recovering a backup system after the database volumes have already been restored**

The LOGONLY keyword in the following control statement indicates that RESTORE SYSTEM is to apply any outstanding log changes to the database. The utility is not to restore the volume copies. In this example, the database volumes were restored outside of Db2. RESTORE SYSTEM applies log changes; it never restores the log copy pool.

```
//STEP1 EXEC DSNUPROC,TIME=1440,
  // UTPROC='',
  // SYSTEM='DSN'
  //SYSIN DD *
  RESTORE SYSTEM LOGONLY
/*
```
Example 3: Recovering a dump on tape of the database copy pool

The following control statement specifies that the RESTORE SYSTEM utility is to consider for restore only dumps on tape of the database copy pool. During the restore, the utility dynamically allocates a maximum of four tape units.

```
//SYSOPRB JOB (ACCOUNT), 'NAME', CLASS='K
//UTIL EXEC DSNUPROC, SYSTEM=V91A, UID='TEMB', UTPROC=''
//*
//*
//DSNUPROC.SYSUT1 DD DSN=SYSOPR.SYSUT1,
// DISP=(MOD,DELETE,CATLG),
// SPACE=(16384,(20,20),,,ROUND),
// UNIT=SYSDA
//DSNUPROC.SYSIN DD *
// RESTORE SYSTEM FROMDUMP TAPEUNITS 4
```

Example 4: Recovering a backup system after the database volumes have already been restored and VCAT aliases renamed

The LOGONLY keyword in the following control statement indicates that RESTORE SYSTEM is to apply any outstanding log changes to the database. The utility is not to restore the volume copies. In this example, the database volumes were restored outside of Db2. RESTORE SYSTEM applies log changes; it never restores the log copy pool. The SWITCH VCAT SYSVALEDDDN(SYSVALUE) keywords indicate that the SYSVALUE DD name data set contains a list of pairs of integrated catalog facility (ICF) (VCAT) aliases. The first (VCAT) alias is the name when the backup was created and the second (VCAT) alias is the name after any renaming is complete. The (VCAT) alias DSNC000 is specified as both the first and second alias since it was not renamed and might be encountered in the log.

```
//STEP1 EXEC DSNUPROC, TIME=1440,
// UTPROC='',
// SYSTEM='DSN'
//SYSIN DD *
// RESTORE SYSTEM LOGONLY SWITCH VCAT SYSVALEDDDN(SYSVALUE)
/*
//SYSVALUE DD *
VCAT1, VCAT2
VCAT5, 21234567
DSNC000, DSNC000
*/
```

Related concepts:

- Point-in-time recovery with system-level backups (Db2 Administration Guide)

Related reference:

- “Syntax and options of the RESTORE SYSTEM control statement” on page 732
- Chapter 8, “BACKUP SYSTEM,” on page 51
Chapter 32. RUNSTATS

The RUNSTATS online utility gathers summary information about the characteristics of data in table spaces, indexes, and partitions. Db2 records these statistics in the Db2 catalog and uses them to select access paths to data during the bind process.

You can use these statistics to evaluate the database design and determine when table spaces or indexes must be reorganized. To obtain the updated statistics, you can query the catalog tables.

The two formats for the RUNSTATS utility are RUNSTATS TABLESPACE and RUNSTATS INDEX. RUNSTATS TABLESPACE gathers statistics on a table space and, optionally, on tables, indexes or columns; RUNSTATS INDEX gathers statistics only on indexes. RUNSTATS does not collect statistics for clone tables or index spaces.

RUNSTATS can collect statistics on any single column or set of columns. RUNSTATS collects the following types of distribution statistics:

**Frequency**
The percentage of rows in the table that contain a value for a column or combination of values for a set of columns.

**Cardinality**
The number of distinct values in the column or set of columns.

When you run RUNSTATS TABLESPACE, you can use the COLGROUP option to collect frequency and cardinality statistics on any column group. You can also collect frequency and cardinality statistics on any single column. When you run RUNSTATS INDEX, you can collect frequency statistics on the leading column of an index and multi-column frequency and cardinality statistics on the leading concatenated columns of an index.

When you run RUNSTATS TABLESPACE, you can use the HISTOGRAM option, with the COLGROUP option, to indicate that histogram statistics are to be gathered for the specified group of columns. RUNSTATS TABLESPACE does not collect histogram statistics for LOB table spaces or XML table spaces. When you run RUNSTATS INDEX, histogram statistics can only be collected on the prefix columns with the same order. Key columns with a mixed order are not allowed for histogram statistics. RUNSTATS INDEX does not collect histogram statistics for XML node ID indexes or XML indexes.

**Restriction:** You cannot run RUNSTATS on directory objects.

**Before running RUNSTATS**

Use caution when running RUNSTATS after any user has manually updated the statistic columns in the catalog. RUNSTATS replaces any values that the user changed.

**Important:** Use care when issuing SQL statements or using tools to update statistics values in catalog tables. If such updates introduce invalid data, unpredictable results can occur, including abends for RUNSTATS and other
utilities. If such problems occur, you can run the RUNSTATS utility and collect statistics at the table space level to resolve the problems, in most cases.

If you plan to use another utility to load or reorganize the same data, consider collecting inline statistics with that utility to avoid the cost of running the RUNSTATS utility separately.

Restriction: RUNSTATS might not provide useful statistics on encrypted data.

Data sets that RUNSTATS uses

See “Data sets that RUNSTATS uses” on page 767

Concurrency and compatibility for RUNSTATS

See “Concurrency and compatibility for RUNSTATS” on page 770

RUNSTATS syntax

RUNSTATS uses different statistics depending on the target objects. For syntax diagrams and options descriptions for RUNSTATS, see:

- “RUNSTATS TABLESPACE syntax and options” on page 743
- “RUNSTATS INDEX syntax and options” on page 758
- “Statistics profile syntax” on page 764

Output

RUNSTATS updates the Db2 catalog with table space or index space statistics, prints a report, or both. For more information, see “Review of RUNSTATS output” on page 776

Important: Use care when issuing SQL statements or using tools to update statistics values in catalog tables. If such updates introduce invalid data, unpredictable results can occur, including abends for RUNSTATS and other utilities. If such problems occur, you can run the RUNSTATS utility and collect statistics at the table space level to resolve the problems, in most cases.

Authorization required

To execute this utility, you must use a privilege set that includes one of the following authorities:

- STATS privilege for the database
- DBADM, DBCTRL, or DBMAINT authority for the database. If the object on which the utility operates is in an implicitly created database, DBADM authority on the implicitly created database or DSNDB04 is required.
- System DBADM authority
- SQLADM authority
- SYSCTRL or SYSADM authority

An ID with installation SYSOPR authority can also execute the RUNSTATS utility, but only on a table space in the DSNDB06 database.

To use RUNSTATS with the REPORT YES option, you must have the SELECT privilege on the reported tables. RUNSTATS does not report values from tables that the user is not authorized to see.
To gather statistics on a LOB table space, you must have SYSADM or DBADM authority for the LOB table space.

**Execution phases of RUNSTATS**

The RUNSTATS utility operates in the following phases:

1. The UTILINIT phase performs initialization.
2. The RUNSTATS phase scans table space or index and updates catalog. If you specify COLGROUP, RUNSTATS also performs a subtask that sorts one or more column group’s data. If you specify FREQVAL with COLGROUP or are collecting frequency statistics for data-partitioned secondary indexes, RUNSTATS also performs a subtask that sorts the partition-level frequency data.
3. The UTILITERM phase cleans up.

**Termination or restart of RUNSTATS**

You can terminate RUNSTATS with the TERM UTILITY command. You can restart a RUNSTATS utility job, but it starts from the beginning again.

**After running RUNSTATS**

After running RUNSTATS with the UPDATE ACCESSPATH, UPDATE SPACE, or the UPDATE ALL options, rebind any application plans that use the tables or indexes so that they use the new statistics.

**Related concepts:**

- Statistics profiles (Db2 Performance)

**Related tasks:**

- Maintaining Db2 database statistics (Db2 Performance)
- Automating statistics maintenance (Db2 Performance)
- Invalidating statements in the dynamic statement cache (Db2 Performance)

**RUNSTATS TABLESPACE syntax and options**

RUNSTATS TABLESPACE utility control statements define operations completed by RUNSTATS utility jobs.

You can create a control statement with the ISPF/PDF edit function. After you create it, save it in a sequential or partitioned data set. When you create the JCL for running the job, use the SYSIN DD statement to specify the name of the data set that contains the utility control statement.

The following syntax and descriptions apply to RUSNTATS TABLESPACE control statements, including use of the INDEX keyword. For the syntax and options of RUNSTATS INDEX control statements, see “RUNSTATS INDEX syntax and options” on page 758.
RUNSTATS TABLESPACE syntax diagram

listdef-spec

listdef-name

table-space-spec

database-name

table-space-name

PART integer

FORCEROLLUP NO

FORCEROLLUP YES

statistics-spec
### Notes:

1. The TABLE keyword is not valid for a LOB table space.
2. You cannot specify INDEX if either USE PROFILE or DELETE PROFILE option is also specified.
3. INDEX(*) is an internal representation of INDEX(ALL) that Db2 uses only in the context of RUNSTATS profiles, and is not valid when specified in any RUNSTATS control statement. When you specify the INDEX(ALL) option in a RUNSTATS control statement that creates a profile, Db2 uses INDEX(*) in the PROFILE_TEXT column of the SYSIBM.SYSTABLES_PROFILES catalog table. However, you must specify INDEX(*) instead of INDEX(ALL)) if you modify the profile by updating the value of the PROFILE_TEXT column directly.

### all-tables-spec

<table>
<thead>
<tr>
<th>TABLE</th>
<th>(ALL)</th>
<th>all-tables-spec</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE</td>
<td>table-name</td>
<td>named-tables-spec</td>
</tr>
</tbody>
</table>

### named-tables-spec

| TABLE | table-name | named-tables-spec |

### correlation-stats-spec

<table>
<thead>
<tr>
<th>INDEX</th>
<th>(ALL)</th>
<th>correlation-stats-spec</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDEX</td>
<td>index-name</td>
<td>correlation-stats-spec</td>
</tr>
</tbody>
</table>

### history-spec

<table>
<thead>
<tr>
<th>STATCLGMEMSRT</th>
<th>integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHRLEVEL</td>
<td>CHANGE</td>
</tr>
<tr>
<td>SHRLEVEL</td>
<td>REFERENCE</td>
</tr>
<tr>
<td>REPORT</td>
<td>NO</td>
</tr>
<tr>
<td>REPORT</td>
<td>YES</td>
</tr>
<tr>
<td>UPDATE</td>
<td>ALL</td>
</tr>
<tr>
<td>ACCESSPATH</td>
<td>SPACE</td>
</tr>
<tr>
<td>ACCESSPATH</td>
<td>NONE</td>
</tr>
</tbody>
</table>

| history-spec | SORTDEV | device-type |

<table>
<thead>
<tr>
<th>sample-spec</th>
<th>use-profile-spec</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE PROFILE</td>
<td>use-profile-spec</td>
</tr>
</tbody>
</table>
Notes:
1 The TABLESAMPLE keyword is only valid for single-table table spaces. Dropped tables are included in this count until REORG, COPY, and MODIFY RECOVERY are run.
2 When using TABLESAMPLE to sample multi-table table spaces or table spaces that are segmented and not partitioned, page sampling is not done and execution continues. The TABLESAMPLE keyword is not valid for a LOB table space.
3 If one type of PROFILE function is specified on one TABLE clause the same type of PROFILE function must be specified on all TABLE clauses.

sample-spec:

use-profile-spec:

Notes:
1 When USE PROFILE is specified with the TABLE (ALL) keywords and no profile exists for a target table, TABLE ALL INDEX ALL is used for the profile specification.
2 When USE PROFILE is specified and no profile exists for a target table, COLUMN ALL INDEX ALL is used for the profile specification.
Notes:
1. The same column name must not be specified more than once. If all columns are listed in the COLUMN option, RUNSTAT will treat it as is, and does not replace it with the COLUMN(ALL) option. Users have to state explicitly if they want the COLUMN(ALL) option.
Notes:
1 The KEYCARD option is deprecated. The KEYCARD functionality is now built into the normal execution of the RUNSTATS INDEX utility and cannot be disabled.

set-profile-spec

Notes:
1 The FROM EXISTING STATS clause is not valid if a column-spec, colgroup-spec, or INDEX keyword has been specified

history-spec

Notes:
1 You can change the default HISTORY value by modifying the STATISTICS HISTORY subsystem parameter. By default, this value is NONE.
RUNSTATS TABLESPACE option descriptions

**TABLESPACE** database-name.table-space-name

Specifies the table space (and, optionally, the database to which it belongs) on which table space and table statistics are to be gathered. This keyword must not identify a table space in DSNDDB01 or work file databases, which consist of DSNDDB07 objects and user-defined work file objects.

**LIST** listdef-name

Specifies the name of a previously defined LISTDEF list name. You can specify one LIST keyword for each RUNSTATS control statement. When you specify this keyword with RUNSTATS TABLESPACE, the list must contain only table spaces. Do not specify LIST with keywords from the TABLE...(table-name) specification. Instead, specify LIST with TABLE (ALL). Likewise, do not specify LIST with keywords from the INDEX...(index-name) specification. You cannot specify index names with a list. Use INDEX(ALL) instead.

If you specify LIST, you cannot specify the PART option. Instead, use the PARTLEVEL option on the LISTDEF statement. The TABLESPACE keyword is required to validate the contents of the list. RUNSTATS TABLESPACE is invoked once for each item in the list.

The partitions or partition ranges can be specified in a list.

**database-name**

Identifies the name of the database to which the table space belongs.

The default value is **DSNDDB04**.

**table-space-name**

Identifies the name of the table space on which statistics are to be gathered.

If the table space that is specified by the TABLESPACE keyword is a LOB table space, you can specify only the following additional keywords: SHRLEVEL REFERENCE or CHANGE, REPORT YES or NO, and UPDATE ALL or NONE.

**PART** integer

Identifies a table space partition on which statistics are to be collected.

**integer** is the physical partition number. It must be in the range from 1 to the number of partitions that are defined for the table space. The maximum is 4096.

You cannot specify PART with LIST.

**TABLE**

Specifies the table on which column statistics are to be gathered. All tables must belong to the table space that is specified in the TABLESPACE option.
You cannot specify the TABLE option for a LOB table space. However, you can specify the TABLE option with the LIST option, even if the specified list includes LOB table spaces. In this case, the TABLE keyword applies to only the non-LOB table spaces and is ignored for the LOB table spaces.

**(ALL)**

Specifies that column statistics are to be gathered on all columns of all tables in the table space. The parentheses around ALL are optional.

The TABLE option value cannot specify a LOB table. However, if TABLE(ALL) is specified, and one or more of the tables in the table space have a LOB column, no error is issued for the LOB tables. RUNSTATS gathers table and column statistics only for the non-LOB tables.

**(table-name)**

Specifies the tables on which column statistics are to be gathered. If you omit the qualifier, RUNSTATS uses the user identifier for the utility job as the qualifier. Enclose the table name in quotation marks if the name contains a blank.

If you specify more than one table, you must repeat the TABLE option. Multiple TABLE options must be specified entirely before or after any INDEX keyword is specified. For example, the INDEX keyword cannot be specified between any two TABLE keywords.

**SAMPLE integer**

Indicates the percentage of rows that RUNSTATS is to sample when collecting statistics on non-leading-indexed columns of an index or non-indexed columns. You can specify any value from 1 through 100.

The default value is 25.

When a low SAMPLE value is specified, inaccuracies in sampling can result in inaccurate statistics and thus bad access paths. The smaller the sample, the more potential for inaccuracy, which can affect access paths.

You cannot specify SAMPLE for LOB table spaces.

**USE PROFILE**

Specifies a stored statistics profile that is used to gather statistics for a table. The statistics profile is created by using the SET PROFILE option and is updated by using the UPDATE PROFILE option.

The column, column group, and index specifications are not allowed as part of the control statement, but are used when stored in the statistics profile.

If no profile exists for the specified table, default statistics are collected:

- When a table name is not specified, TABLE ALL INDEX ALL is used for the profile specification.
- When a table name is specified, COLUMN ALL INDEX ALL is used for the profile specification.

**INCLUDE NPI or INCLUDE (NPI)**

Specifies that statistics are to be collected on the non-partitioned indexes that listed in the profile for the table. The INCLUDE NPI clause is not valid if the PART keyword is not specified at the RUNSTATS TABLESPACE level. You must specify the INCLUDE and NPI keywords together. The parentheses around NPI are optional.

**DELETE PROFILE**

Specifies an existing RUNSTATS profile that is to be deleted from the SYSIBM.SYSTABLES_PROFILES catalog table.
Column, column group, and index specifications are not allowed as part of the control statement when the DELETE PROFILE option is used. No statistics are collected when you specify this option in the RUNSTATS control statement.

**TABLESAMPLE SYSTEM**

Allows RUNSTATS to collect statistics on a sample of the data pages from the table.

System sampling considers each page individually, including that page with probability P/100 (where P is the value of numeric-literal) and excluding it with probability 1-P/100. Unless the optional REPEATABLE clause is specified, each execution of RUNSTATS usually yields a different such sample of the table. The size of the sample is controlled by the integer parameter in parentheses, representing an approximate percentage P of the table to be returned. Only a percentage of the data pages as specified through the numeric-literal parameter is retrieved and used for the statistics collection. This keyword is valid only for single-table table spaces.

If you run RUNSTATS with the TABLESAMPLE SYSTEM AUTO keywords for an entire partitioned table space, and the number of rows in each partition varies greatly, a partition that contains rows on very few pages can be skipped. In such cases, a DSNU1375I message indicates each partition that is not sampled. It is best to run RUNSTATS again with the SAMPLE keyword at the partition level for each such partition.

The TABLESAMPLE keyword is not valid for a LOB table space.

When RUNSTATS TABLESPACE TABLE is run on a multi-table table space or a table space that is segmented and not partitioned, the TABLESAMPLE keyword is not used. Instead, all pages are scanned to collect statistics.

**numeric-literal**

Specifies the size of the sample to be obtained, as a percentage P. This value must be a positive number that is less than or equal to 100 and greater than 0. For example, a value of 0.01 represents one one-hundredth of a percent, such that 1 row in 10,000 would be sampled, on average. A value greater than 100, zero, or a value less than zero is treated by Db2 as an error. The smallest allowable positive number for this option is 0.01 percent.

Depending on table space size and sampling rate that is used, it is possible that a partition is not included in the sample. In this case, RUNSTATS does not collect statistics for this partition, and might report warnings or errors for aggregate statistics.

When `numeric-literal` is specified, and real-time statistics are not available, Db2 issues message DSNU3343I.

**AUTO**

When "AUTO" is specified, RUNSTATS determines a sampling rate based on the size of the table when RUNSTATS runs. The larger the table the smaller the sampling rate. The threshold for sampling is when the table has more than 500,000 rows; otherwise all pages are read. The same threshold is applicable for TABLESPACE sampling with PART option specified. The number of rows is obtained from the real-time statistics report.

When AUTO is specified, and real-time statistics are not available, RUNSTATS sets the sampling rate to 100.
REPEATABLE integer
Adding the REPEATABLE clause to the TABLESAMPLE clause ensures that repeated executions of RUNSTATS return the same sample. The integer parameter is a non-negative integer that represents the seed to be used in sampling. Passing a negative seed results in an error (DSNU048I). The sample set might still vary between repeatable RUNSTATS invocations. Variations can occur if activity against the table or statistical view results in changes to the table or statistical view data since the last time TABLESAMPLE REPEATABLE was run.

SET PROFILE
Specifies that RUNSTATS generates a RUNSTATS profile for the specified table from the options that are specified in the current RUNSTATS invocation. RUNSTATS stores the profile in the SYSIBM.SYSTABLES_PROFILES catalog table. No statistics are collected when you specify this option in the RUNSTATS control statement. For more information about the options that you can specify in a profile, and the syntax for specifying the options, see: “Statistics profile syntax” on page 764.

FROM EXISTING STATS
Specifies that RUNSTATS generate a statistics profile with options that are based on analysis of the statistics that currently exist for the specified table. This option can be specified only with the SET PROFILE option. The keywords used in the generated profile do not necessarily match those used to collect the statistics previously.

UPDATE PROFILE
Specifies that RUNSTATS updates an existing statistics profile in the SYSIBM.SYSTABLES_PROFILES catalog table with the options specified in the current RUNSTATS control statement. No statistics are collected when you specify this option in the RUNSTATS control statement. If the column or COLGROUP specification already exists in the profile, the new specification replaces the existing one.

COLUMN
Specifies columns on which column statistics are to be gathered.
You can specify this option only if you specify a particular table on which statistics are to be gathered. (Use the TABLE (table-name) option to specify a particular table.) If you specify particular tables and do not specify the COLUMN option, RUNSTATS uses the default, COLUMN(ALL). If you do not specify a particular table with the TABLE option, you cannot specify the COLUMN option; however, in this case, COLUMN(ALL) is assumed.

(ALL)
Specify that statistics are to be gathered on all columns in the table.
The COLUMN (ALL) option is not allowed for LOB table spaces.

(column-name, ...)
Specify the columns on which statistics are to be gathered. You can specify a list of column names. If you specify more than one column, separate each name with a comma.
The more columns that you specify, the longer the job takes to complete.

COLGROUP (column-name, ...)
Indicates that the specified set of columns are to be treated as a group. This option enables RUNSTATS to collect a cardinality value on the specified column group. RUNSTATS TABLESPACE ignores COLGROUP when processing XML table spaces and indexes.
When you specify the COLGROUP keyword, RUNSTATS collects correlation statistics for the specified column group. If you want RUNSTATS to also collect distribution statistics, specify the FREQVAL option with COLGROUP.

(column-name, ...) specifies the names of the columns that are part of the column group.

When you define a column group on a single column, you can potentially improve RUNSTATS performance by specifying the STATCLGMEMSRT option or changing the value of the STATCLGSRT subsystem parameter. You can use these options to avoid the external data sort that is used for column group processing.

To specify more than one column group, repeat the COLGROUP option.

Restrictions:
- The length of a COLGROUP value cannot exceed the maximum length of the COLVALUE column in the SYSIBM.SYSCOLDIST catalog table.
- A RUNSTATS control statement can contain a maximum of 255 COLGROUP specifications.

Related information:

| STATISTICS COLGROUP DATA SORT STG LIMIT field (STATCLGSRT subsystem parameter) (Db2 Installation and Migration) |

**FREQVAL**
Indicates, when specified with the COLGROUP option, that frequency statistics are also to be gathered for the specified group of columns. (COLGROUP indicates that cardinality statistics are to be gathered.) One group of statistics is gathered for each column. RUNSTATS TABLESPACE ignores FREQVAL MOST, FREQVAL LEAST, or FREQVAL BOTH when it processes XML table spaces and indexes.

**COUNT** integer
Specifies how many frequently occurring values are collected from the specified column group. You must specify a value for integer. If the COUNT keyword is not specified, the default is COUNT 10 MOST

It is best to specify a COUNT value that is not greater than the value of COLCARDF minus one, for the column group. For most situations, 10 is usually a reasonable value. Greater COUNT values might be needed to detect skewed data, especially in high cardinality cases. However, avoid values greater than 100 in most cases. Specifying a value of 1000 or more can increase the prepare time for some SQL statements.

**MOST**
Indicates that the utility collects the most frequently occurring values for the specified set of columns when COLGROUP is specified. For example, FREQVAL COUNT 10 MOST means that the 10 most frequently occurring values are collected.

**BOTH**
Indicates that the utility collects the most and the least frequently occurring values for the specified set of columns when COLGROUP is specified. If COUNT is n, the utility collects the n least frequently occurring values and the n most frequently occurring values.

**LEAST**
Indicates that the utility collects the least frequently occurring values for the specified set of columns when COLGROUP is specified.
HISTOGRAM
Indicates, when specified with the COLGROUP (see colgroup-stats-spec) option of RUNSTATS TABLESPACE, that histogram statistics are to be gathered for the specified group of columns. RUNSTATS TABLESPACE ignores HISTOGRAM when processing XML table spaces and indexes.

NUMQUANTILES integer
Indicates the number of quantiles that the utility collects. The integer value must be greater than or equal to one. The number of quantiles that you specify should never exceed the total number of distinct values in the column or the column group. The maximum number of quantiles that is allowed is 100.
When the NUMQUANTILES keyword is omitted, NUMQUANTILES takes a default value of 100. Based on the number of records in the table, the number of quantiles is readjusted down to an optimal number.

INDEX
Specifies indexes on which statistics are to be gathered. RUNSTATS gathers column statistics for the first column of the index, and depending on the options that you specify, possibly additional index columns. All the indexes must be associated with the same table space, which must be the table space that is specified in the TABLESPACE option.
INDEX can be used on auxiliary tables to gather statistics on an index.

(ALL)
Specifies that column statistics are to be gathered for all indexes that are defined on tables that are contained in the table space.

(*)
Specifies that statistics are to be gathered for all indexes that are defined on the specified table.

(index-name, ...)
Specifies the indexes for which statistics are to be gathered. You can specify a list of index names. If you specify more than one index, separate each name with a comma. Enclose the index name in quotation marks if the name contains a blank.

PART integer
Identifies an index partition on which statistics are to be collected.

integer is the number of the partition.

KEYCARD
The KEYCARD option is deprecated in the RUNSTATS TABLESPACE control statement and no longer needs to be specified to collect statistics on the values in the key columns of an index if INDEX is specified.
The RUNSTATS utility automatically collects all of the distinct values in all of the 1 to \( n \) intermediate key column combinations for the specified indexes, where \( n \) is the number of columns in the index. For example, suppose that you have an index defined on three columns: A, B, and C. RUNSTATS collects cardinality statistics for column A, column set A and B, and column set A, B, and C. With the deprecation of KEYCARD, this functionality cannot be disabled.
The RUNSTATS utility tolerates the specification of the KEYCARD option. The utility does not issue any messages if the control statement includes or excludes the KEYCARD option when INDEX is specified.
FREQVAL
Controls, when specified with the INDEX option, the collection of frequent-value statistics. If you specify FREQVAL with INDEX, this keyword must be followed by the NUMCOLS keyword.

NUMCOLS integer
Indicates the number of columns in the index for which RUNSTATS collects frequently occurring values. integer can be a number between 1 and the number of indexed columns. If you specify a number greater than the number of indexed columns, RUNSTATS uses the number of columns in the index.

For example, suppose that you have an index defined on three columns: A, B, and C. If you specify NUMCOLS 1, Db2 collects frequently occurring values for column A. If you specify NUMCOLS 2, Db2 collects frequently occurring values for the column set A and B. If you specify NUMCOLS 3, Db2 collect frequently occurring values for the column set A, B, and C.

The default value is 1, which means that RUNSTATS collects frequently occurring values on the first key column of the index.

COUNT integer
Indicates the number of frequently occurring values that are to be collected from the specified key columns. For example, specifying 15 means that RUNSTATS is to collect 15 frequently occurring values from the specified key columns.

The default value is 10.

HISTOGRAM
Indicates, when specified with the INDEX option (correlation-stats-spec) for RUNSTATS TABLE SPACE, that histogram statistics are to be gathered for the specified key columns. Histogram statistics can be collected only on the prefix columns with the same order. Key columns for histogram statistics with a mixed order are not allowed.

When RUNSTATS collects histogram statistics for partition table spaces, it aggregates them into SYSCOLDIST.

NUMQUANTILES integer
Indicates how many quantiles that the utility is to collect. The integer value must be greater than or equal to one. The number of quantiles that you specify should never exceed the total number of distinct values in the key columns specified. The maximum number of quantiles that is allowed is 100.

When the NUMQUANTILES keyword is omitted, NUMQUANTILES takes a default value of 100. Based on the number of keys in the index, the number of quantiles is readjusted down to an optimal number.

STATCLGMEMSRT integer
Specifies the amount of memory that the utility can use for sorting records when collecting statistics on a single column that is defined with the COLGROUP option. Use STATCLGMEMSRT to avoid column group sorts by an external sort program, which can negatively affect the performance of RUNSTATS.

integer specifies the number of megabytes of memory space that the utility can use for an in-memory sort. Valid values are 0 to 4096. If the amount of space that is needed for the sort exceeds the integer value, the utility invokes a sort program. If you specify 0, the utility automatically invokes a sort program.
The amount of space that is needed for the column group sort depends on the following factors:

- The number of column groups for which RUNSTATS is collecting statistics
- The length of the single-column column group
- The number of distinct values in the column (cardinality)

The value of STATCLGMEMSR override the value of the STATCLGSRT subsystem parameter.

**Related information:**

[STATISTICS COLGROUP DATA SORT STG LIMIT field (STATCLGSRT subsystem parameter) (Db2 Installation and Migration)]

**SHRLEVEL**

Indicates whether other programs that access the table space while RUNSTATS is running must use read-only access or can change the table space.

**CHANGE**

Allows other programs to change the table space or index. With SHRLEVEL CHANGE, RUNSTATS might collect statistics on uncommitted data.

**REFERENCE**

Allows only read-only access by other programs.

**REPORT**

Specifies whether RUNSTATS is to generate a set of messages that report the collected statistics.

**NO**

Indicates that RUNSTATS is not to generate the set of messages.

**YES**

Indicates that the set of messages is to be sent as output to SYSPRINT. The messages that RUNSTATS generates depend on the combination of keywords in the utility control statement. However, these messages do not depend on the value of the UPDATE option. REPORT YES always generates a report of space and access path statistics.

**UPDATE**

Indicates which collected statistics are to be inserted into the catalog tables.

**ALL**

Indicates that all collected statistics are to be updated in the catalog.

**ACCESSPATH**

Indicates that Db2 is to update the catalog with only those statistics that are used for access path selection.

**SPACE**

Indicates that Db2 is to update the catalog with only space-related statistics.

**NONE**

Indicates that no catalog tables are to be updated with the collected statistics.

Running the RUNSTATS always invalidates the dynamic cache. However, when you specify UPDATE NONE REPORT NO, RUNSTATS invalidates statements in the dynamic statement cache without collecting statistics, updating catalogs tables, or generating reports.
HISTORY
  Indicates which statistics are to be recorded in the catalog history tables. The
  value that you specify for HISTORY does not depend on the value that you
  specify for UPDATE.

  The default is the value of the STATISTICS HISTORY subsystem parameter on
  the DSNTIPO installation panel. By default, this parameter value is NONE.

ALL
  Indicates that all collected statistics are to be updated in the catalog history
  tables.

ACCESSPATH
  Indicates that Db2 is to update the catalog history tables with only those
  statistics that are used for access path selection.

SPACE
  Indicates that Db2 is to update the catalog history tables with only
  space-related statistics.

NONE
  Indicates that no catalog history tables are to be updated with the collected
  statistics.

SORTDEVT
  Specifies the device type that the sort program uses to dynamically allocate the
  sort work data sets that are required.

  device-type
    Specifies any disk device type that is acceptable for the DYNALLOC
    parameter of the SORT or OPTIONS option of the external sort program.
    Tape devices are not supported by the sort program.

If all of the following conditions are true, SORTDEVT defaults to SYSALLDA
and the temporary data sets are dynamically allocated:
  • You omit SORTDEVT.
  • A sort is required.
  • You did not provide the DD statements that the sort program requires for
    the temporary data sets

If you specify SORTDEVT and omit SORTNUM, no value is passed to the sort
program; the sort program uses its own default.

SORTNUM
  Specifies the number of required sort work data sets that the sort program is to
  allocate.

  integer is the number of temporary data sets that can range from 2 to 255.

You need at least two sort work data sets for each sort. The SORTNUM value
applies to each sort invocation in the utility. For example, suppose RUNSTATS
is being run on three indexes, SORTKEYS is specified, no constraints limit
parallelism, and SORTNUM is specified as 8. In this case, a total of 24 sort
work data sets are allocated for a job.

Each sort work data set uses both above the line and below the line virtual
storage. Therefore, if you specify too high a value for SORTNUM, the utility
might decrease the degree of parallelism due to virtual storage constraints, and
possibly decrease the degree down to one, which means that no parallelism is
used.
**Important:** The SORTNUM keyword is ignored if the UTSORTAL subsystem parameter is set to YES and the IGNSORTN subsystem parameter is set to YES.

**FORCEROLLUP**
Specifies whether aggregation or roll up of statistics is to occur even if statistics were not gathered on some partitions. This option enables the optimizer to select the best access path.

**YES**
Indicates that forced aggregation or rollup processing is to be done, even though some partitions might not contain data.

**NO**
Indicates that aggregation or rollup is to be done only if data is available for all partitions.

If the value for STATISTICS ROLLUP on panel DSNTIPO is NO and data is not available for all partitions, Db2 issues message DSNU623I.

**RESET ACCESSPATH**
Resets access path statistics for all tables in the specified table space and related indexes. Real-time statistics and space statistics in the catalog for the target objects are not reset. For a complete list of the statistics that are reset or deleted when you specify this option, see: “Resetting access path statistics” on page 780.

**Important:** You cannot recover previous values after the RUNSTATS utility is invoked with the RESET ACCESSPATH option, unless a statistics history is maintained. Specifying the HISTORY_ACCESSPATH option only records when the access path statistics were reset, and does not provide a method for recovering the previous values. For more information about how to maintain a statistics history, see “Collecting statistics history” on page 773.

Statements that refer to the objects for which statistics are reset are invalidated in the dynamic statement cache.

This option cannot be specified for LOB table spaces.

When this RESET ACCESSPATH is used, other keywords that specify the specific statistics to be collected within the table space cannot be specified.

**HISTORY ACCESSPATH**
Inserts rows into the following tables for each object for which the access path statistics are reset when the RESET ACCESSPATH option is specified:
- SYSTABLES_HIST for tables.
- SYSINDEXES_HIST for indexes.

Related tasks:
- [Automating statistics maintenance](#)

Related reference:
- Chapter 18, “LISTDEF,” on page 215
- [Db2 Sort for z/OS](#)

Related information:
- [DFSORT Application Programming Guide](#)

---

**RUNSTATS INDEX syntax and options**

RUNSTATS INDEX utility control statements define the operations completed by RUNSTATS utility jobs.
You can create a control statement with the ISPF/PDF edit function. After you create it, save it in a sequential or partitioned data set. When you create the JCL for running the job, use the SYSIN DD statement to specify the name of the data set that contains the utility control statement.

The following syntax and descriptions apply to RUSNTATS INDEX control statements. For the syntax and options of RUNSTATS TABLESPACE control statements, including use of the INDEX keyword, see "RUNSTATS TABLESPACE syntax and options" on page 743.

**RUNSTATS INDEX syntax diagram**

```
RUNSTATS INDEX

LIST listdef-name correlation-stats-spec

(index-name PART integer correlation-stats-spec)

(ALL) TABLESPACE tablespace-name correlation-stats-spec
database-name.

SHRLEVEL CHANGE REPORT NO UPDATE ALL

SHRLEVEL REFERENCE REPORT YES UPDATE ACCESSPATH SPACE

REPORT NO UPDATE ACCESSPATH SPACE

SORTDEVT device-type

SORTNUM integer HISTORY NONE FORCEROLLUP NO

HISTORY ALL ACCESSPATH SPACE FORCEROLLUP YES
```

**Notes:**
1. You can change the default HISTORY value by modifying the STATISTICS HISTORY subsystem parameter. By default, this value is NONE.

**correlation-stats-spec:**
Notes:
1 The KEYCARD option is deprecated. The KEYCARD functionality is now built into the normal execution of the RUNSTATS INDEX utility and cannot be disabled.

RUNSTATS INDEX option descriptions

INDEX
Specifies the indexes on which statistics are to be gathered. Column statistics are gathered on the first column of the index. All of the indexes must be associated with the same table space.

LIST listdef-name
Specifies the name of a previously defined LISTDEF list name. You can specify one LIST keyword for each RUNSTATS control statement. When you specify LIST with RUNSTATS INDEX, the list must contain only index spaces. Do not specify LIST with keywords from the INDEX...(index-name) specification; except for the correlation-stats-spec.

RUNSTATS groups indexes by their related table space. RUNSTATS INDEX is invoked once per table space. The INDEX keyword is required to validate the contents of the LIST.

(index-name, ...)
Specifies the indexes on which statistics are to be gathered. You can specify a list of index names. If you specify more than one index, separate each name with a comma. Enclose the index name in quotation marks if the name contains a blank.

PART integer
Identifies the index partition on which statistics are to be collected.

integer is the number of the partition.

(ALL)
Specifies that statistics are to be gathered on all indexes that are defined on all tables in the specified table space.

TABLESPACE
Identifies the table space and, optionally, the database to which it belongs, for which index statistics are to be gathered.

database-name
The name of the database to which the table space belongs.

The default value is DSNDB04.
The name of the table space for which index statistics are to be gathered.

**KEYCARD**

The KEYCARD option is deprecated in the RUNSTATS INDEX control statement and no longer needs to be specified to collect statistics on the values in the key columns of an index.

Except when processing XML NODEID or XML values indexes, the RUNSTATS utility automatically collects all of the distinct values in all of the 1 to \( n \) key column combinations for the specified indexes, where \( n \) is the number of columns in the index. For example, suppose that you have an index defined on three columns: A, B, and C. RUNSTATS collects cardinality statistics for column A, column set A and B, and column set A, B, and C. With the deprecation of KEYCARD, this functionality cannot be disabled.

The RUNSTATS utility tolerates the specification of the KEYCARD option. The utility does not issue any messages if the control statement includes or excludes the KEYCARD option when INDEX is specified.

**FREQVAL**

Controls, when specified with the INDEX option, the collection of frequent-value statistics. If you specify FREQVAL with INDEX, this keyword must be followed by the NUMCOLS keyword. RUNSTATS INDEX ignores FREQVAL MOST/LEAST/BOTH when processing XML NODEID or XML values indexes.

**NUMCOLS** integer

Indicates the number of columns in the index for which RUNSTATS is to collect frequently occurring values. integer can be a number between 1 and the number of indexed columns. If you specify a number greater than the number of indexed columns, RUNSTATS uses the number of columns in the index.

For example, suppose that you have an index defined on three columns: A, B, and C. If you specify NUMCOLS 1, Db2 collects frequently occurring values for column A. If you specify NUMCOLS 2, Db2 collects frequently occurring values for the column set A and B. If you specify NUMCOLS 3, Db2 collects frequently occurring values for the column set A, B, and C.

The default value is 1, which means that RUNSTATS is to collect frequently occurring values on the first key column of the index.

**COUNT** integer

Indicates the number of frequently occurring values that are to be collected from the specified key columns. For example, specifying 15 means that RUNSTATS is to collect 15 frequently occurring values from the specified key columns.

The default is 10.

**MOST**

Indicates that the utility is to collect the most frequently occurring values for the specified set of key columns when FREQVAL NUMCOLS COUNT MOST keywords are specified.

**LEAST**

Indicates that the utility is to collect the least frequently occurring values for the specified set of key columns when FREQVAL NUMCOLS COUNT LEAST keywords are specified.
**BOTH**
Indicates that the utility is to collect the most and the least frequently occurring values for the specified set of key columns when FREQVAL NUMCOLS COUNT BOTH keywords are specified.

**HISTOGRAM**
Indicates, when specified with the INDEX option (see correlation-stats-spec) for RUNSTATS INDEX, that histogram statistics are to be gathered for the specified key columns. Histogram statistics can be collected only on the prefix columns with the same order. Key columns for histogram statistics with a mixed order are not allowed.

When RUNSTATS collects histogram statistics for partitioned indexes, it aggregates them into SYSCOLDIST. RUNSTATS INDEX ignores the HISTOGRAM keyword when processing XML NODEID or XML values indexes.

**NUMQUANTILES**  integer
Indicates how many quantiles that the utility is to collect. The integer value must be greater than or equal to one. The number of quantiles that you specify should never exceed the total number of distinct values in the specified key columns. The maximum number of quantiles is 100.

When the NUMQUANTILES keyword is omitted, NUMQUANTILES takes a default value of 100. Based on the number of keys in the index, the number of quantiles is readjusted down to an optimal number.

**SHRLEVEL**
Indicates whether other programs that access the table space while RUNSTATS is running must use read-only access or can change the table space.

**CHANGE**
Allows other programs to change the table space or index. With SHRLEVEL CHANGE, RUNSTATS might collect statistics on uncommitted data.

**REFERENCE**
Allows only read-only access by other programs.

**REPORT**
Specifies whether RUNSTATS is to generate a set of messages that report the collected statistics.

**NO**
Indicates that RUNSTATS is not to generate the set of messages.

**YES**
Indicates that the set of messages is to be sent as output to SYSPRINT. The messages that RUNSTATS generates depend on the combination of keywords in the utility control statement. However, these messages do not depend on the value of the UPDATE option. REPORT YES always generates a report of space and access path statistics.

**UPDATE**
Indicates which collected statistics are to be inserted into the catalog tables.

**ALL**
Indicates that all collected statistics are to be updated in the catalog.

**ACCESSPATH**
Indicates that Db2 is to update the catalog with only those statistics that are used for access path selection.
**SPACE**
Indicates that Db2 is to update the catalog with only space-related statistics.

**NONE**
Indicates that no catalog tables are to be updated with the collected statistics.

Running RUNSTATS always invalidates the dynamic cache. However, when you specify UPDATE NONE REPORT NO, RUNSTATS invalidates statements in the dynamic statement cache without collecting statistics, updating catalogs tables, or generating reports.

**SORTDEVT**
Specifies the device type that the external sort program uses to dynamically allocate the sort work data sets that are required.

`device-type`
Specifies any disk device type that is acceptable for the DYNALLOC parameter of the SORT or OPTIONS option of the external sort program.

If all of the following conditions are true, SORTDEVT defaults to SYSALLDA and the temporary data sets are dynamically allocated:

- You omit SORTDEVT.
- A sort is required.
- You did not provide the DD statements that the sort program requires for the temporary data sets

If you specify SORTDEVT and omit SORTNUM, no value is passed to the sort program; the sort program uses its own default.

**SORTNUM**
Specifies the number of required sort work data sets that the sort program is to allocate.

`integer` is the number of temporary data sets that can range from 2 to 255.

You need at least two sort work data sets for each sort. The SORTNUM value applies to each sort invocation in the utility. For example, suppose that RUNSTATS is running on three indexes, SORTKEYS is specified, no constraints limit parallelism, and SORTNUM is specified as 8. In this case, a total of 24 sort work data sets are allocated for a job.

Each sort work data set uses both above the line and below the line virtual storage. Therefore, if you specify too high a value for SORTNUM, the utility might decrease the degree of parallelism due to virtual storage constraints, and possibly decrease the degree down to one, which means that no parallelism is used.

**HISTORY**
Indicates which statistics are to be recorded in the catalog history tables. The value that you specify for HISTORY does not depend on the value that you specify for UPDATE.

The default is the value of the STATISTICS HISTORY subsystem parameter on the DSNTIPO installation panel. By default, this parameter value is NONE.

**ALL**
Indicates that all collected statistics are to be updated in the catalog history tables.
ACCESSPATH
Indicates that Db2 is to update the catalog history tables with only those statistics that are used for access path selection.

SPACE
Indicates that Db2 is to update the catalog history tables with only space-related statistics.

NONE
Indicates that no catalog history tables are to be updated with the collected statistics.

FORCEROLLUP
Specifies whether aggregation or rollup of statistics is to occur even if statistics were not gathered on some partitions. This option enables the optimizer to select the best access path.

YES
Indicates that forced aggregation or rollup processing is to be done, even though some partitions might not contain data.

NO
Indicates that aggregation or rollup is to be done only if data is available for all partitions.

If the value for STATISTICS ROLLUP on panel DSNTIPO is NO and data is not available for all partitions, Db2 issues message DSNU623I.

Related tasks:
- Automating statistics maintenance (Db2 Performance)

Related reference:
- Chapter 18, “LISTDEF,” on page 215
- Db2 Sort for z/OS

Related information:
- DFSORT Application Programming Guide

Statistics profile syntax

The options of a statistics profile are stored within the PROFILE_TEXT column of the SYSTABLES_PROFILES catalog table.

You can specify the following statistics collection options in a statistics profile:
- COLUMN
- COLGROUP
- FREQVAL
- COUNT
- MOST
- BOTH
- LEAST
- INDEX
- KEYCARD
- NUMCOLS
- COUNT
- MOST
• BOTH
• LEAST
• HISTOGRAM
• NUMQUANTILES

The profile contains the default values for any options that are not specified.

When you update an existing profile that contains a partitioned index, the PART keyword must be specified on all index specifications for that index, or omitted from the index specification for that index. Statistics profile processing enforces this requirement. Any profile modifications done through SQL statements must follow the same restriction, or error messages result when the profile is used.

For a given partitioned index:
• Any new index specifications without the PART keyword replace all index specifications in the profile regardless of the PART keyword specification.
• Any new index specification with the PART keyword replaces only the existing index specification with the same PART specified, or a specification without the PART keyword.

The PROFILE functions cannot be executed when there are syntax errors in the statistics profile. Syntax errors can be corrected using RUNSTATS UPDATE PROFILE or SQL UPDATE, or by deleting the profile with RUNSTATS DELETE PROFILE or SQL DELETE.

```
column-spec
  colgroup-spec  index-spec
```

column-spec:
```
COLUMN(ALL), COLUMN(column-name)
```

Notes:
1. The same column name must not be specified more than once. If all columns are listed in the COLUMN option, RUNSTATS treats it as-is, and does not replace the list with the COLUMN(ALL) option. You must specify the COLUMN(ALL) option explicitly.

```
colgroup-spec:
```

index-spec:

Notes:
1 INDEX(*) is an internal representation of INDEX(ALL) that Db2 uses only in the context of RUNSTATS profiles, and is not valid when specified in any RUNSTATS control statement. When you specify the INDEX(ALL) option in a RUNSTATS control statement that creates a profile, Db2 uses INDEX(*) in the PROFILE_TEXT column of the SYSIBM.SYSTABLES_PROFILES catalog table. However, you must specify INDEX(*) instead of INDEX(ALL)) if you modify the profile by updating the value of the PROFILE_TEXT column directly.

colgroup-stats-spec:

correlation-stats-spec:
Notes:

1 The KEYCARD option is deprecated. The KEYCARD functionality is now built into the normal execution of the RUNSTATS utility and cannot be disabled.

Statistics profile options

Statistics profile options have the same meanings as they do when specified directly in a RUNSTATS utility control statement. For more information, see “RUNSTATS TABLESPACE syntax and options” on page 743.

Related tasks:
- Automating statistics maintenance (Db2 Performance)
- Maintaining Db2 database statistics (Db2 Performance)
- Updating statistics profiles (Db2 Performance)

Related information:

Data sets that RUNSTATS uses

The RUNSTATS utility uses a number of data sets during its operation.

The following table lists the data sets that RUNSTATS uses. The table lists the DD name that is used to identify the data set, a description of the data set, and an indication of whether it is required. Include statements in your JCL for each required data set and any optional data sets that you want to use.

Table 105. Data sets that RUNSTATS uses

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSIN</td>
<td>Input data set that contains the utility control statement.</td>
<td>Yes</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>Output data set for messages.</td>
<td>Yes</td>
</tr>
<tr>
<td>RNPRIN01</td>
<td>A data set that contains messages from the sort program (usually, SYSOUT or DUMMY). This data set is used when distribution statistics are collected for column groups.</td>
<td>No</td>
</tr>
</tbody>
</table>
Table 105. Data sets that RUNSTATS uses (continued)

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>STPRIN01</td>
<td>A data set that contains messages from the sort program (usually, SYSOUT or DUMMY). This data set is used when frequency statistics are collected on data-partitioned secondary indexes, or when TABLESPACE TABLE COLGROUP FREQVAL is specified.</td>
<td>Yes</td>
</tr>
<tr>
<td>Sort work data sets</td>
<td>Temporary data sets for sort input and output when collecting statistics on at least one data-partitioned secondary index. This data set is used when the COLGROUP option is specified or the COLGROUP and FREQVAL options are specified. The DD names have the form ST01WKnn.</td>
<td>No</td>
</tr>
<tr>
<td>Sort work data sets</td>
<td>Temporary data sets for sort input and output when collecting distribution statistics for column groups. The DD names have the form STATWK01.</td>
<td>No</td>
</tr>
<tr>
<td>Sort work data sets</td>
<td>Temporary data sets for sort input and output when collecting frequency statistics. The DD names have the form SORTWK01 and ST02WKnn.</td>
<td>No</td>
</tr>
</tbody>
</table>

**Note:**
1. Required when collecting distribution statistics for column groups.
2. STPRIN01 is required if statistics are being collected on at least one data-partitioned secondary index, but RUNSTATS dynamically allocates the STPRIN01 data set if UTPRINT is allocated to SYSOUT.
3. Required when collecting statistics on at least one data-partitioned secondary index.
4. If the DYNALLOC parm of the SORT program is not turned on, you need to allocate the data set. Otherwise, the sort program dynamically allocates the temporary data set.
5. Required when the COLGROUP with FREQVAL options are specified.
6. It is recommended that you use dynamic allocation by specifying SORTDEVT in the utility statement because dynamic allocation reduces the maintenance required of the utility job JCL.

The following objects are named in the utility control statement and do not require DD statements in the JCL:

**Table space or index**
Object that is to be scanned.

**Calculating the size of the sort work data sets**

Depending on the type of statistics that RUNSTATS collects, the utility uses the ST01WKnn data sets, the SORTWK01 data set, both types of data sets, or neither.

The ST01WKnn data sets are used when collecting statistics on at least one data-partitioned secondary index. To calculate the approximate size (in bytes) of the ST01WKnn data set, use the following formula:

\[ 2 \times (\text{maximum record length} \times \text{numcols} \times (\text{count} + 2) \times \text{number of indexes}) \]

The variables in the preceding formula have the following values:
maximum record length
Maximum record length of the SYSCOLDISTSTATS record that is processed when collecting frequency statistics (You can obtain this value from the RECLENGTH column in SYSTABLES.)

numcols
Number of key columns to concatenate when you collect frequent values from the specified index.

count
Number of frequent values that RUNSTATS is to collect.

The SORTWK01 data set is used when collecting distribution statistics. To calculate the approximate size (in bytes) of the SORTWK01 data set, use the following formula:

\[(\text{longest record length} + \text{prefix}) \times \text{sum from 1 to } N (\#\text{colgroups}_n \times \#\text{rows} - n)\]

The variables in the preceding formula have the following values:

\(N\) Number of tables for which distribution statistics are collected
\(\#\text{colgroups}_n\) Number of column groups that are specified for the nth table
\(\#\text{rows}\) Number of rows for the nth table

The ST02WKmn data sets are used when collecting frequency statistics on at least one COLGROUP. To calculate the approximate size (in bytes) of the ST02WKmn data set, use the following formula:

\[2 \times (\text{maximum record length} \times (\text{count} + 2) \times \text{number of parts})\]

The variables in the preceding formula have the following values:

maximum record length
Maximum record length of the SYSCOLDISTSTATS record that is processed when collecting frequency statistics (You can obtain this value from the RECLENGTH column in SYSTABLES.)

count
Number of frequent values that RUNSTATS is to collect.

Sort work data sets cannot span volumes. Smaller volumes require more sort work data sets to sort the same amount of data; therefore, large volume sizes can reduce the number of needed sort work data sets. When you allocate sort work data sets on disk, the recommended amount of space to allow provides at least 1.2 times the amount of data that is to be sorted.

Tape devices are not supported for sort work data sets.

Related concepts:
“Data sets that online utilities use” on page 16

Related reference:
Db2 Sort for z/OS

Related information:
DFSORT Application Programming Guide
Concurrency and compatibility for RUNSTATS

The RUNSTATS utility has certain concurrency and compatibility characteristics associated with it.

Db2 treats individual data and index partitions as distinct target objects. Utilities operating on different partitions of the same table space or index space are compatible.

Claims

The following table shows which claim classes RUNSTATS claims and drains and any restrictive state that the utility sets on the target object.

Table 106. Claim classes of RUNSTATS operations

<table>
<thead>
<tr>
<th>Target</th>
<th>RUNSTATS TABLESPACE SHRLEVEL REFERENCE</th>
<th>RUNSTATS TABLESPACE SHRLEVEL CHANGE</th>
<th>RUNSTATS INDEX SHRLEVEL REFERENCE</th>
<th>RUNSTATS INDEX SHRLEVEL CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table space or partition</td>
<td>DW/UTRO</td>
<td>CR/UTRW(^1)</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Index or partition</td>
<td>None</td>
<td>None</td>
<td>DW/UTRO</td>
<td>CR/UTRW</td>
</tr>
</tbody>
</table>

**Legend:**
- DW - Drain the write claim class - concurrent access for SQL readers.
- CR - Claim the read claim class.
- UTRO - Utility restrictive state - read-only access allowed.
- UTRW - Utility restrictive state - read-write access allowed.
- None - Object is not affected by this utility.

**Note:**
1. If the target object is a segmented table space, SHRLEVEL CHANGE does not allow you to concurrently execute an SQL searched DELETE without the WHERE clause.

Compatibility

The following table shows which utilities can run concurrently with RUNSTATS on the same target object. The target object can be a table space, an index space, or a partition of a table space or index space. If compatibility depends on particular options of a utility, that information is also shown in the table.

Table 107. Compatibility of RUNSTATS with other utilities

<table>
<thead>
<tr>
<th>Utility</th>
<th>RUNSTATS TABLESPACE SHRLEVEL REFERENCE</th>
<th>RUNSTATS TABLESPACE SHRLEVEL CHANGE</th>
<th>RUNSTATS INDEX SHRLEVEL REFERENCE</th>
<th>RUNSTATS INDEX SHRLEVEL CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHECK DATA DELETE NO</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CHECK DATA DELETE YES</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CHECK INDEX</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CHECK LOB</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COPY INDEXSPACE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COPY TABLESPACE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DIAGNOSE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Utility</td>
<td>RUNSTATS TABLESPACE SHRLEVEL REFERENCE</td>
<td>RUNSTATS TABLESPACE SHRLEVEL CHANGE</td>
<td>RUNSTATS INDEX SHRLEVEL REFERENCE</td>
<td>RUNSTATS INDEX SHRLEVEL CHANGE</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------</td>
<td>---------------------------------------</td>
<td>-------------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>LOAD</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LOAD SHRLEVEL CHANGE</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>MERGECOPY</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MODIFY RECOVERY</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>QUIESCE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>REBUILD INDEX</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RECOVER ERROR RANGE</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RECOVER INDEX</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RECOVER INDEX TOCOPY or TOLOGPOINT</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RECOVER TABLESPACE (no options)</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RECOVER TABLESPACE TOCOPY or TORBA</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REORG INDEX</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REORG TABLESPACE UNLOAD CONTINUE or PAUSE</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REORG TABLESPACE UNLOAD ONLY or EXTERNAL</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>REPAIR DUMP or VERIFY</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>REPAIR LOCATE INDEX PAGE REPLACE</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REPAIR LOCATE KEY or RID DELETE or REPLACE</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>REPAIR LOCATE TABLESPACE PAGE REPLACE</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>REPORT</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RUNSTATS</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>STOPSPACE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UNLOAD</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Collecting distribution statistics for column groups**

When RUNSTATS collects distribution statistics for column groups, the utility invokes a sort program to sort the distribution statistics. This sort requires its own work data set. The DD name is STATWK01.

**About this task**

You can let this data set be dynamically allocated through the sort program, or you can allocate the data set through a DD statement in the job JCL.
If you need to control the size or placement of the data sets, use the JCL statements to allocate STATWK01.

If the column group contains one column, you can potentially improve RUNSTATS performance by specifying the STATCLGMEMSRT option or changing the value of the STATCLGSR option. You can use these options to avoid the external data sort that is used for column group processing.

**Procedure**

To collect distribution statistics for column groups:

- To let the work data set be dynamically allocated, remove the STATWK01 DD statements from the job and allocate the UTPRINT statement to SYSOUT.
- To let the sort program dynamically allocate this data set, specify the SORTDEV option in the RUNSTATS utility control statement.

**Related reference:**

- "Data sets that RUNSTATS uses" on page 767
- [Db2 Sort for z/OS](#)
- [STATISTICS COLGROUP DATA SORT STG LIMIT field (STATCLGSR subsystem parameter) (Db2 Installation and Migration)](#)

**Related information:**

- [DFSORT Application Programming Guide](#)

---

**Updating statistics for a partitioned table space**

You can run RUNSTATS on one or more single partitions of one or more table spaces or indexes (including data-partitioned secondary indexes). When you run the utility on a single partition of an object, RUNSTATS uses the resulting partition-level statistics to update the aggregate statistics for the entire object.

For partition-by-growth table spaces, RUNSTATS waits to drain the table space or index if necessary. If the object does not drain, RUNSTATS continues trying to drain the object. However, RUNSTATS does not have its own options to control this drain behavior as some other utilities do. (Other utilities have the DRAIN_WAIT and RETRY options). Instead, RUNSTATS uses the IRLMRWT subsystem parameter value for the drain wait time and the UTIMOUT subsystem parameter value for the retry value. If RUNSTATS finds these values to be excessive, it uses a lower value.

**Related reference:**

- [RESOURCE TIMEOUT field (IRLMRWT subsystem parameter) (Db2 Installation and Migration)](#)
- [UTILITY TIMEOUT field (UTIMOUT subsystem parameter) (Db2 Installation and Migration)](#)

---

**Collection of statistics on the Db2 catalog and directory**

You can use the RUNSTATS utility to gather statistics for the Db2 catalog. Db2 uses the collected statistics on the catalog to determine the access path for user queries of the catalog. You cannot run RUNSTATS on any Db2 directory objects.
The following sample shows part of the output from a RUNSTATS job on a catalog table space and its indexes:

```
DSNU000I  DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = DSNTEX
DSNU050I  DSNUGUTC - RUNSTATS TABLESPACE DSNDB06.SYSDBASE INDEX(ALL)
DSNU610I  # DSNUSUTP - SYSTABLEPART CATALOG UPDATE FOR DSNDB06.SYSDBASE SUCCESSFUL
DSNU610I  # DSNUSUTS - SYSTABLESPACE CATALOG UPDATE FOR DSNDB06.SYSDBASE SUCCESSFUL
DSNU610I  # DSNUSUTB - SYSTABLES CATALOG UPDATE FOR SYSIBM.SYSTABLESPACE SUCCESSFUL
DSNU610I  # DSNUSUTB - SYSTABLES CATALOG UPDATE FOR SYSIBM.SYSSYNONYMS SUCCESSFUL
DSNU610I  # DSNUSUIX - SYSINDEXES CATALOG UPDATE FOR SYSIBM.DSNDSX01 SUCCESSFUL
DSNU610I  # DSNUSUPIP - SYSINDEXPART CATALOG UPDATE FOR SYSIBM.DSNDSX01 SUCCESSFUL
DSNU610I  # DSNUSUCO - SYSCOLUMNS CATALOG UPDATE FOR SYSIBM.DSNDSX01 SUCCESSFUL
DSNU610I  # DSNUSUFL - SYSFIELDS CATALOG UPDATE FOR SYSIBM.DSNDSX01 SUCCESSFUL
DSNU610I  # DSNUSUIX - SYSINDEXES CATALOG UPDATE FOR SYSIBM.DSNDSY01 SUCCESSFUL
DSNU610I  # DSNUSUPIP - SYSINDEXPART CATALOG UPDATE FOR SYSIBM.DSNDSY01 SUCCESSFUL
DSNU610I  # DSNUSUCO - SYSCOLUMNS CATALOG UPDATE FOR SYSIBM.DSNDSY01 SUCCESSFUL
DSNU610I  # DSNUSUFL - SYSFIELDS CATALOG UPDATE FOR SYSIBM.DSNDSY01 SUCCESSFUL
DSNU010I  DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```

Figure 76. Example RUNSTATS output from a job on a catalog table space

## Collecting frequency statistics for data-partitioned secondary indexes

When RUNSTATS collects frequency statistics on at least one data-partitioned secondary index, the utility invokes a sort program to sort the statistics. This sort requires temporary sort work data sets. The DD name is ST01WKnn.

### About this task

You can let the ST01WKnn data sets be dynamically allocated through the SORT program or allocate the data sets through DD statements in the job JCL. If you need to control the size or placement of the data sets, use the JCL statements to allocate ST01WKnn.

### Procedure

To collect frequency statistics for data-partitioned secondary indexes:
- To let the sort work data sets be dynamically allocated, remove the ST01WKnn DD statements from the job and allocate the UTPRINT statement to SYSOUT.
- To let the SORT program dynamically allocate these data sets, specify the SORTDEV option in the RUNSTATS utility control statement to specify the device type for the temporary data sets. Optionally, you can also use the SORTNUM option to specify the number of temporary data sets to use.

### Related reference:

- ["Data sets that RUNSTATS uses” on page 767](#)
- [Db2 Sort for z/OS](#)

### Related information:

- [DFSORT Application Programming Guide](#)

## Collecting statistics history

You can collect statistics history by using the RUNSTATS utility.
Procedure

To collect statistics history:

Specify the HISTORY option in the RUNSTATS utility control statement. When you specify HISTORY with a value other than NONE, RUNSTATS updates the catalog history tables with the access path statistics, space statistics, or both, depending on the parameter that you specify with HISTORY. The HISTORY option does not update the main catalog statistics that Db2 uses to select access paths. You can use the HISTORY option to monitor how statistics change over time without updating the main catalog statistics that Db2 uses to select access paths.

Related tasks:

- Collecting history statistics (Db2 Performance)

Related reference:

- History statistics (Db2 Performance)
- History statistics (Db2 Performance)
- Catalog Table Name
  - Column Name
    - SYSCOLSTATS
      - COLCARD: The number of distinct values in the XML column. This value is set to -2 for other types of XML objects.
      - HIGHKEY: The highest value of the column. This value is set to blanks for other types of XML objects.
      - HIGH2KEY: The second highest value of the column. This value is set to blanks for other types of XML objects.
      - LOW2KEY: The second lowest value of the column. This value is set to blanks for other types of XML objects.
      - LOWKEY: The lowest value of the column. This value is set to blanks for other types of XML objects.

Collection of statistics on XML objects

You can use separate RUNSTATS control statements to collect statistics on XML table spaces, or on their associated base table spaces.

You can specify that RUNSTATS collects space statistics on an XML table space. You can use those statistics to determine when the XML table space must be reorganized. Statistics that are collected on the XML table space also affect access path selection.

The following tables shows statistics values that are collected for the implicitly created XML DOCID columns. The

<table>
<thead>
<tr>
<th>Catalog Table Name</th>
<th>Column Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSCOLSTATS</td>
<td>COLCARD</td>
<td>The number of distinct values in the XML column. This value is set to -2 for other types of XML objects.</td>
</tr>
<tr>
<td>SYSCOLSTATS</td>
<td>HIGHKEY</td>
<td>The highest value of the column. This value is set to blanks for other types of XML objects.</td>
</tr>
<tr>
<td>SYSCOLSTATS</td>
<td>HIGH2KEY</td>
<td>The second highest value of the column. This value is set to blanks for other types of XML objects.</td>
</tr>
<tr>
<td>SYSCOLSTATS</td>
<td>LOW2KEY</td>
<td>The second lowest value of the column. This value is set to blanks for other types of XML objects.</td>
</tr>
<tr>
<td>SYSCOLSTATS</td>
<td>LOWKEY</td>
<td>The lowest value of the column. This value is set to blanks for other types of XML objects.</td>
</tr>
</tbody>
</table>
Table 108. Statistics that are collected on the XML DOCID column. (continued)

<table>
<thead>
<tr>
<th>Catalog Table Name</th>
<th>Column Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSCOLUMNS</td>
<td>COLCARDF</td>
<td>Estimated number of distinct values in the column. This value is set to -2 for other types of XML objects.</td>
</tr>
<tr>
<td>SYSCOLUMNS</td>
<td>HIGH2KEY</td>
<td>Second highest value in the column. This value is set to blanks for other types of XML objects.</td>
</tr>
<tr>
<td>SYSCOLUMNS</td>
<td>LOW2KEY</td>
<td>Second lowest value in the column. This value is set to blanks for other types of XML objects.</td>
</tr>
</tbody>
</table>

The following table shows that statistics values that are collected when RUNSTATS is run for XML indexes:

Table 109. Statistics that RUNSTATS collects for XML indexes

<table>
<thead>
<tr>
<th>Catalog Table Name</th>
<th>Column Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSINDEXES</td>
<td>FIRSTKEYCARDF</td>
<td>The number of unique key values in the index.</td>
</tr>
<tr>
<td>SYSINDEXES</td>
<td>FULLKEYCARDF</td>
<td>The total number of key entries in the index.</td>
</tr>
<tr>
<td>SYSINDEXES</td>
<td>FIRSTKEYCARDF</td>
<td>The number of documents in the XML column. This value is collected only for implicitly created node ID indexes.</td>
</tr>
<tr>
<td>SYSINDEXES</td>
<td>CLUSTERRATIOF</td>
<td>This value is set to -2.</td>
</tr>
<tr>
<td>SYSKEYTARGETS</td>
<td>HIGH2KEY</td>
<td>For KEYSEQ=1, the second-highest key value.</td>
</tr>
<tr>
<td>SYSKEYTARGETS</td>
<td>LOW2KEY</td>
<td>For KEYSEQ=1, the second-lowest key value.</td>
</tr>
<tr>
<td>SYSKEYTARGETS</td>
<td>CARDF</td>
<td>The number of unique DOCID values in the index.</td>
</tr>
<tr>
<td>SYSKEYTARGETS</td>
<td>KEYCARDF</td>
<td>This value is set to -2.</td>
</tr>
</tbody>
</table>

RUNSTATS ignores the following keywords for certain types of objects in XML table spaces:
- COLGROUP
- KEYCARD (for indexes)
- FREQVAL MOST | LEAST | BOTH
- HISTOGRAM

DB2 issues message DSNU1345I when these keywords are ignored for an XML object.

XML indexes are related to XML tables, and not to the associated base tables. If you specify a base table space and an XML index in the same RUNSTATS control statement, Db2 generates an error. When you run RUNSTATS against a base table, RUNSTATS collects statistics only for indexes on the base table, including the document ID index.

Related concepts:
- Filter factor estimation for the XMLEXISTS predicate (Db2 Performance)
- Storage structure for XML data (Db2 Programming for XML)
- XML data indexing (Db2 Programming for XML)
- XML data and query performance (Db2 Performance)
- Best practices for XML performance in Db2 (Db2 Performance)

Related information:
- DSNU1354I (Db2 Messages)
Review of RUNSTATS output

The RUNSTATS utility updates columns in the catalog tables. When you specify REPORT YES, the RUNSTATS utility also generates a report of the statistics that it gathered.

The following table shows the catalog tables that RUNSTATS updates depending on the value of the UPDATE option, the value of the HISTORY option, and the source of the statistics (table space, partition, index or LOB table space).

Table 110. Catalog tables that RUNSTATS updates

<table>
<thead>
<tr>
<th>Keyword</th>
<th>UPDATE option</th>
<th>HISTORY option</th>
<th>Catalog table that RUNSTATS updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLESPACE</td>
<td>UPDATE ALL</td>
<td>HISTORY ALL</td>
<td>SYSTABLESPACE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSTABLEPART</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSTABLEPART_HIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSTABLES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSTABLES_HIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSTABSTATS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSTABSTATS_HIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSLOBSTATS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSLOBSTATS_HIST</td>
</tr>
<tr>
<td>TABLESPACE</td>
<td>UPDATE ALL</td>
<td>HISTORY ACCESSPATH</td>
<td>SYSTABLESPACE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSTABLES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSTABLES_HIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSTABSTATS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSTABSTATS_HIST</td>
</tr>
<tr>
<td>TABLESPACE</td>
<td>UPDATE ALL</td>
<td>HISTORY SPACE</td>
<td>SYSTABLEPART</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSTABLEPART_HIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSTABLES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSTABLES_HIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSTABSTATS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSTABSTATS_HIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSLOBSTATS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSLOBSTATS_HIST</td>
</tr>
<tr>
<td>TABLESPACE</td>
<td>UPDATE ACCESSPATH</td>
<td>HISTORY ALL</td>
<td>SYSTABLESPACE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSTABLES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSTABLES_HIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSTABSTATS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSTABSTATS_HIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSLOBSTATS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSLOBSTATS_HIST</td>
</tr>
<tr>
<td>TABLESPACE</td>
<td>UPDATE ACCESSPATH</td>
<td>HISTORY ACCESSPATH</td>
<td>SYSTABLESPACE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSTABLES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSTABLES_HIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSTABSTATS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSTABSTATS_HIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSLOBSTATS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSLOBSTATS_HIST</td>
</tr>
<tr>
<td>TABLESPACE</td>
<td>UPDATE ACCESSPATH</td>
<td>HISTORY SPACE</td>
<td>none</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>none</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>none</td>
</tr>
<tr>
<td>TABLESPACE</td>
<td>UPDATE SPACE</td>
<td>HISTORY ALL</td>
<td>SYSTABLEPART</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSTABLEPART_HIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSLOBSTATS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSLOBSTATS_HIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSTABLES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSTABLES_HIST</td>
</tr>
<tr>
<td>TABLESPACE</td>
<td>UPDATE SPACE</td>
<td>HISTORY ACCESSPATH</td>
<td>none</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>none</td>
</tr>
<tr>
<td>TABLESPACE</td>
<td>UPDATE SPACE</td>
<td>HISTORY SPACE</td>
<td>none</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>none</td>
</tr>
<tr>
<td>Keyword</td>
<td>UPDATE option</td>
<td>HISTORY option</td>
<td>Catalog table that RUNSTATS updates</td>
</tr>
<tr>
<td>---------</td>
<td>---------------</td>
<td>----------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>TABLE</td>
<td>UPDATE ALL</td>
<td>HISTORY ALL  4</td>
<td>SYSCOLUMNS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSCOLSTATS2</td>
</tr>
<tr>
<td>TABLE</td>
<td>UPDATE ALL</td>
<td>HISTORY ACCESSPATH</td>
<td>SYSCOLUMNS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSCOLSTATS2</td>
</tr>
<tr>
<td>TABLE</td>
<td>UPDATE ALL</td>
<td>HISTORY SPACE</td>
<td>none</td>
</tr>
<tr>
<td>TABLE</td>
<td>UPDATE ACCESSPATH</td>
<td>HISTORY ALL  4</td>
<td>SYSCOLUMNS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSCOLSTATS2</td>
</tr>
<tr>
<td>TABLE</td>
<td>UPDATE ACCESSPATH</td>
<td>HISTORY ACCESSPATH</td>
<td>SYSCOLUMNS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSCOLSTATS2</td>
</tr>
<tr>
<td>TABLE</td>
<td>UPDATE ACCESSPATH</td>
<td>HISTORY SPACE</td>
<td>none</td>
</tr>
<tr>
<td>INDEX</td>
<td>UPDATE ALL</td>
<td>HISTORY ALL  4</td>
<td>SYSCOLUMNS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSCOLSTATS2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXES_HIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXPART</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXPART_HIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXSTATS2</td>
</tr>
<tr>
<td>INDEX</td>
<td>UPDATE ALL</td>
<td>HISTORY ACCESSPATH</td>
<td>SYSCOLUMNS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSCOLSTATS2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXES_HIST</td>
</tr>
<tr>
<td>INDEX</td>
<td>UPDATE ALL</td>
<td>HISTORY SPACE</td>
<td>SYSCOLSTATS2</td>
</tr>
<tr>
<td>INDEX</td>
<td>UPDATE ACCESSPATH</td>
<td>HISTORY ALL  4</td>
<td>SYSCOLSTATS2</td>
</tr>
<tr>
<td>INDEX</td>
<td>UPDATE ACCESSPATH</td>
<td>HISTORY ACCESSPATH</td>
<td>SYSCOLSTATS2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXES_HIST</td>
</tr>
<tr>
<td>INDEX</td>
<td>UPDATE ACCESSPATH</td>
<td>HISTORY ACCESSPATH</td>
<td>SYSCOLSTATS2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXES_HIST</td>
</tr>
<tr>
<td>Keyword</td>
<td>UPDATE option</td>
<td>HISTORY option</td>
<td>Catalog table that RUNSTATS updates</td>
</tr>
<tr>
<td>---------</td>
<td>---------------</td>
<td>----------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>INDEX</td>
<td>UPDATE SPACE</td>
<td>HISTORY ALL (^{4})</td>
<td>SYSINDEXPART</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXPART_HIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXES(^{3})</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXES_HIST(^{5})</td>
</tr>
<tr>
<td>INDEX</td>
<td>UPDATE SPACE</td>
<td>HISTORY ACCESSPATH</td>
<td>none</td>
</tr>
<tr>
<td>INDEX</td>
<td>UPDATE SPACE</td>
<td>HISTORY SPACE</td>
<td>SYSINDEXPART</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXES(^{3})</td>
</tr>
<tr>
<td>INDEX(^{6})</td>
<td>UPDATE ALL</td>
<td>HISTORY ALL(^{4})</td>
<td>SYSKEYTARGETS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSKEYTARGETS_HIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSKEYTARGETSTATS(^{3})</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSKEYTGTDISTSTATS(^{3})</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXPART</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXPART_HIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXSTATS(^{2})</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXSTATS_HIST(^{2})</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXES_HIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSKEYTGTDIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSKEYTGTDIST_HIST</td>
</tr>
<tr>
<td>INDEX(^{6})</td>
<td>UPDATE ALL</td>
<td>HISTORY ACCESSPATH</td>
<td>SYSKEYTARGETS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSKEYTARGETS_HIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSKEYTARGETSTATS(^{3})</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSKEYTGTDISTSTATS(^{3})</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXPART</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXPART_HIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXSTATS(^{2})</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXSTATS_HIST(^{2})</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXES_HIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSKEYTGTDIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSKEYTGTDIST_HIST</td>
</tr>
<tr>
<td>INDEX(^{6})</td>
<td>UPDATE ALL</td>
<td>HISTORY SPACE</td>
<td>SYSKEYTARGETS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSKEYTARGETSTATS(^{3})</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSKEYTGTDISTSTATS(^{3})</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXPART</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXPART_HIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXSTATS(^{2})</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXSTATS_HIST(^{2})</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXES_HIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSKEYTGTDIST</td>
</tr>
<tr>
<td>INDEX(^{6})</td>
<td>UPDATE ACCESSPATH</td>
<td>HISTORY ALL(^{4})</td>
<td>SYSKEYTARGETS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSKEYTARGETS_HIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSKEYTARGETSTATS(^{3})</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSKEYTGTDISTSTATS(^{3})</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXPART</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXPART_HIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXSTATS(^{2})</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXSTATS_HIST(^{2})</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSINDEXES_HIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSKEYTGTDIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SYSKEYTGTDIST_HIST</td>
</tr>
<tr>
<td>Keyword</td>
<td>UPDATE option</td>
<td>HISTORY option</td>
<td>Catalog table that RUNSTATS updates</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------</td>
<td>----------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>INDEX⁶</td>
<td>UPDATE ACCESSPATH</td>
<td>HISTORY ACCESSPATH</td>
<td>SYSKEYTARGETS &lt;br&gt; SYSKEYTARGETSTATS² &lt;br&gt; SYSKEYTGTDISTSTATS² &lt;br&gt; SYSINDEXPART &lt;br&gt; SYSINDEXSTATS² &lt;br&gt; SYSINDEXSTATS_HIST² &lt;br&gt; SYSINDEXES &lt;br&gt; SYSINDEXES_HIST &lt;br&gt; SYSKEYTGTDIST &lt;br&gt; SYSKEYTGTDIST_HIST</td>
</tr>
<tr>
<td>INDEX⁶</td>
<td>UPDATE ACCESSPATH</td>
<td>HISTORY SPACE</td>
<td>SYSKEYTARGETS &lt;br&gt; SYSKEYTARGETSTATS² &lt;br&gt; SYSKEYTGTDISTSTATS² &lt;br&gt; SYSINDEXPART &lt;br&gt; SYSINDEXPART_HIST &lt;br&gt; SYSINDEXSTATS² &lt;br&gt; SYSINDEXES &lt;br&gt; SYSINDEXES_HIST &lt;br&gt; SYSKEYTGTDIST</td>
</tr>
<tr>
<td>INDEX⁶</td>
<td>UPDATE SPACE</td>
<td>HISTORY ALL⁴</td>
<td>SYSKEYTARGETS_HIST &lt;br&gt; SYSINDEXPART &lt;br&gt; SYSINDEXSTATS_HIST² &lt;br&gt; SYSINDEXES &lt;br&gt; SYSINDEXES_HIST &lt;br&gt; SYSKEYTGTDIST_HIST</td>
</tr>
<tr>
<td>INDEX⁶</td>
<td>UPDATE SPACE</td>
<td>HISTORY ACCESSPATH</td>
<td>SYSINDEXPART &lt;br&gt; SYSINDEXES &lt;br&gt; SYSINDEXES_HIST</td>
</tr>
<tr>
<td>INDEX⁶</td>
<td>UPDATE SPACE</td>
<td>HISTORY SPACE</td>
<td>SYSINDEXPART &lt;br&gt; SYSINDEXPART_HIST &lt;br&gt; SYSINDEXES &lt;br&gt; SYSINDEXES_HIST</td>
</tr>
</tbody>
</table>

**Note:**

1. Not applicable if the specified table space is a LOB table space.
2. Only updated for partitioned objects. When you run RUNSTATS against single partitions of an object, RUNSTATS uses the partition-level statistics to update the aggregate statistics for the entire object. These partition-level statistics are contained in the following catalog tables:
   - SYSCOLSTATS
   - SYSCOLDISTSTATS
   - SYSTABSTATS
   - SYSINDEXSTATS
3. Applicable only when the specified table space is a LOB table space.
4. When HISTORY NONE is specified, none of the catalog history tables are updated.
5. Only the SPACEF and STATSTIME columns are updated.
6. Applicable only when the target object is an expression-based index.

RUNSTATS sets the following columns to -1 for universal table spaces and table spaces that are defined as LARGE:
- CARD in SYSTABLES
- CARD in SYSINDEXPART
- FAROFFPOS in SYSINDEXPART
Resetting access path statistics

You can use the RUNSTATS utility to remove old and out-of-date access path statistics for Db2 objects.

About this task

When the RUNSTATS utility is invoked over a period of time, statistics are collected incrementally for target objects. The combination of many changes to target objects and many RUNSTATS invocations, perhaps with different options, might result in some previously collected statistics becoming outdated. Such out-of-date statistics might cause Db2 to choose inefficient access paths for SQL statements. One solution is to invoke the RUNSTATS utility again to refresh the statistics. However, the task of formulating RUNSTATS invocations to solve the problem might prove difficult because of the complicated nature of the many previous RUNSTATS invocations.

Such stale statistics can also increase your statistics collection costs, especially after migration to Db2 12, if their collection becomes standardized in a statistics profile that is based on existing statistics. For more information see [Cleaning stale or unneeded Db2 catalog statistics (Db2 Performance)](https://www.ibm.com/support/knowledgecenter/en/SSPHDQ_12.1.0/com.ibm.db2.luw.admin.performance.doc/doc/com.ibm.db2.luw.admin.performance.doc/r/db2clean.html).

When the situation occurs, you can invoke the RUNSTATS utility to reset the access path statistics for all tables and indexes in a specified table space. When you reset the statistics, the default values are used. No statistics are gathered or reported. Space statistics and real-time statistics are not reset for the specified objects. After your reset access path statistics, the previous values cannot be recovered if no statistics history is available.

Procedure

To reset access path statistics:

- Invoke the RUNSTATS utility, and specify the following options:
  1. Specify the RESET ACCESSPATH option.
  2. Optional: Specify the HISTORY ACCESSPATH option to record that the access path statistics were reset in rows in the SYSIBM.SYSTABLES_HIST and SYSIBM.SYSINDEXES_HIST statistics tables. This option only records that the reset occurred and does not save the access path statistics values that are reset.

For example, you might issue the following utility control statement:

```sql
RUNSTATS TABLESPACE db-name.ts-name RESET ACCESSPATH
```

Statistics are not collected. Instead, the RUNSTATS utility resets the access path statistics.
Results

Certain catalog table rows are updated with default values, and rows are deleted from other catalog tables. All updated rows in the catalog tables contain the same timestamp value. Real-time statistics and space for the specified object are not reset. However, the dynamic statement cache is invalidated.

The following statistics are reset to the specified values:

**SYSIBM.SYSTABLESPACE**

The following values are changed:

<table>
<thead>
<tr>
<th>Column</th>
<th>Changed value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NACTIVE</td>
<td>-1</td>
</tr>
<tr>
<td>NACTIVEF</td>
<td>-1</td>
</tr>
<tr>
<td>STATTIME</td>
<td>The TIMESTAMP value for the reset operation</td>
</tr>
</tbody>
</table>

**SYSIBM.SYSCOLUMNS**

The following values are changed:

<table>
<thead>
<tr>
<th>Column</th>
<th>Changed value</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLCARD</td>
<td>-1</td>
</tr>
<tr>
<td>COLCARDF</td>
<td>-1</td>
</tr>
<tr>
<td>HIGH2KEY</td>
<td>Zero-length blank</td>
</tr>
<tr>
<td>LOW2KEY</td>
<td>Zero-length blank</td>
</tr>
<tr>
<td>STATTIME</td>
<td>The TIMESTAMP value for the reset operation</td>
</tr>
<tr>
<td>STATS_FORMAT</td>
<td>Blank</td>
</tr>
</tbody>
</table>

**SYSIBM.SYSTABLES**

The following values are changed:

<table>
<thead>
<tr>
<th>Column</th>
<th>Changed value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CARD</td>
<td>-1</td>
</tr>
<tr>
<td>CARDF</td>
<td>-1</td>
</tr>
<tr>
<td>NPAGES</td>
<td>-1</td>
</tr>
<tr>
<td>NPAGESF</td>
<td>-1</td>
</tr>
<tr>
<td>PCTPAGES</td>
<td>-1</td>
</tr>
<tr>
<td>PCTROWCOMP</td>
<td>-1</td>
</tr>
<tr>
<td>STATTIME</td>
<td>The TIMESTAMP value for the reset operation</td>
</tr>
</tbody>
</table>

**SYSIBM.SYSINDEXES**

The following values are changed:

<table>
<thead>
<tr>
<th>Column</th>
<th>Changed value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLUSTERED</td>
<td>'N'</td>
</tr>
<tr>
<td>NLEVELS</td>
<td>-1</td>
</tr>
<tr>
<td>NLEAF</td>
<td>-1</td>
</tr>
<tr>
<td>Column</td>
<td>Changed value</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>FIRSTKEYCARD</td>
<td>-1</td>
</tr>
<tr>
<td>FULLKEYCARD</td>
<td>-1</td>
</tr>
<tr>
<td>FIRSTKEYCARDFS</td>
<td>-1</td>
</tr>
<tr>
<td>FULLKEYCARDFS</td>
<td>-1</td>
</tr>
<tr>
<td>CLUSTERRATIO</td>
<td>0</td>
</tr>
<tr>
<td>CLUSTER RATIO FS</td>
<td>0</td>
</tr>
<tr>
<td>DATAREPEATFACTORF</td>
<td>-1</td>
</tr>
<tr>
<td>STATETIME</td>
<td>The TIMESTAMP value for the reset operation</td>
</tr>
</tbody>
</table>

**SYSIBM.SYSKEYTARGETS**

The following values are changed:

<table>
<thead>
<tr>
<th>Column</th>
<th>Changed value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CARDF</td>
<td>-1</td>
</tr>
<tr>
<td>HIGH2KEY</td>
<td>Zero-length blank</td>
</tr>
<tr>
<td>LOW2KEY</td>
<td>Zero-length blank</td>
</tr>
<tr>
<td>STATETIME</td>
<td>TIMESTAMP</td>
</tr>
<tr>
<td>STATS_FORMAT</td>
<td>Blank</td>
</tr>
</tbody>
</table>

Applicable rows are deleted from the following catalog tables for the specified objects:
- SYSIBM.SYSTABSTATS
- SYSIBM.SYSCOLSTATS
- SYSIBM.SYSINDEXSTATS
- SYSIBM.SYCOLDIST
- SYSIBM.SYCOLDISTSTATS
- SYSIBM.SYSKEYTARGETSTATS
- SYSIBM.SYSKEYTGTDIST
- SYSIBM.SYSKEYTGTDISTSTATS

**What to do next**

After resetting the access path statistics objects, collect your standard statistics for those objects. For more information, see Collecting statistics by using Db2 utilities (Db2 Performance).

**Related tasks:**
- [Maintaining Db2 database statistics (Db2 Performance)](link)
- [Improving filter factors by collecting cardinality and frequency statistics (Db2 Performance)](link)
- [Collecting history statistics (Db2 Performance)](link)

**Related reference:**
- [Statistics used for access path selection (Db2 Performance)](link)
Sample RUNSTATS control statements

Use the sample control statements as models for developing your own RUNSTATS control statements.

Example 1: Updating catalog statistics for a table space while allowing changes

The following control statement specifies that the RUNSTATS utility is to update the catalog with statistics for table space DSN8D81A.DSN8S11E and all of its associated tables and indexes. When updating the table statistics, RUNSTATS is to sample 25% of the rows. Although SHRLEVEL CHANGE is not specified, by default Db2 permits other processes to make changes to the table space while the RUNSTATS utility is executing.

```sql
//STEP1 EXEC DSNUPROC,UID='IUJQU225.RUNSTA',TIME=1440,
    //UTPROC='',
    //SYSTEM='DSN'
//UTPRINT DD SYSOUT=* //SYSIN DD *
RUNSTATS TABLESPACE DSN8D81A.DSN8S11E
    TABLE(ALL) SAMPLE 25
    INDEX(ALL)
```

Example 2: Updating index statistics

The following control statement specifies that RUNSTATS is to update the catalog statistics for index DSN8810.XEMPL1.

```sql
RUNSTATS INDEX (DSN8810.XEMPL1)
```

Example 3: Updating index statistics while prohibiting updates

The following control statement specifies that RUNSTATS is to update the catalog statistics for indexes XEMPL1 and XEMPL2. Db2 does not permit other processes to change the table space that is associated with XEMPL1 and XEMPL2 (table space DSN8S11E) while this utility is executing. This restricted access is the default behavior.

```sql
RUNSTATS INDEX (DSN8B10.XEMPL1,DSN8B10.XEMPL2)
```

Example 4: Updating statistics for columns in several tables

The following control statement specifies that RUNSTATS is to update the catalog statistics for the following columns in table space DSN8D11P.DSN8S11C:

- All columns in the TCONA and TOPTVAL tables
- The LINENO and DSPLINE columns in the TDSPTXT table

```sql
RUNSTATS TABLESPACE(DSN8D11P.DSN8S11C)
    TABLE(TCONA)
    TABLE(TOPTVAL) COLUMN(ALL)
    TABLE(TDSPTXT) COLUMN(LINENO,DSPLINE)
```

Example 5: Updating all statistics for a table space

The following control statement specifies that RUNSTATS is to update all catalog statistics (table space, tables, columns, and indexes) for table space DSN8D81P.DSN8S81C.

```sql
RUNSTATS TABLESPACE(DSN8D81P.DSN8S81C) TABLE INDEX
```
Example 6: Updating statistics that are used for access path selection and generating a report

The following control statement specifies that RUNSTATS is to update the catalog with only the statistics that are collected for access path selection. The utility is to report all statistics for the table space and route the report to SYSPRINT.

RUNSTATS TABLESPACE DSN8D11A.DSN8S11E
  REPORT YES
  UPDATE ACCESSPATH

Example 7: Updating all statistics and generating a report

The following control statement specifies that RUNSTATS is to update the catalog with all statistics (access path and space) for table space DSN8D81A.DSN8S81E. The utility is also to report the collected statistics and route the report to SYSPRINT.

RUNSTATS TABLESPACE DSN8D11A.DSN8S11E
  REPORT YES
  UPDATE ALL

Example 8: Reporting statistics without updating the catalog

The following control statement specifies that RUNSTATS is to collect statistics for table space DSN8D81A.DSN8S81E and route the report to SYSPRINT. The utility is not to update the catalog with the collected statistics.

RUNSTATS TABLESPACE DSN8D11A.DSN8S11E
  REPORT YES
  UPDATE NONE

Example 9: Updating statistics for a partition

The following control statement specifies that RUNSTATS is to update the statistics for the first partition of table space DSN8D81A.DSN8S81E and the first partition of the DSN8810.XEMP1 index.

RUNSTATS TABLESPACE DSN8D11A.DSN8S11E PART 1 INDEX(DSN8B10.XEMP1 PART 1)

Example 10: Updating catalog and history tables and reporting all statistics

The following control statement specifies that RUNSTATS is to update the catalog tables and history catalog tables with all statistics for table space DB0E0101.TL0E0101 (including related indexes and columns). The utility is to report the collected statistics and route the statistics to SYSPRINT.

RUNSTATS TABLESPACE DB0E0101.TL0E0101 INDEX TABLE
  REPORT YES
  UPDATE ALL
  HISTORY ALL

Example 11: Updating statistics on frequently occurring values

Assume that the SYSADM.IXNP1 index is defined on four columns: NP1, NP2, NP3, and NP4. The following control statement specifies that RUNSTATS is to update the statistics for index SYSADM.IXNP1.
The RUNSTATS utility collects cardinality statistics for column NP1, column set NP1 and NP2, and column set NP1, NP2, NP3, and NP4. The FREQVAL option and its associated parameters indicate that RUNSTATS is also to collect the 5 most frequently occurring values on column NP1 (the first key column of the index), and the 10 most frequently occurring values on the column set NP1 and NP2 (the first two key columns of the index). The utility is to report the collected statistics and route the statistics to SYSPRINT.

```
RUNSTATS INDEX (SYSADM.IXNPI)
   FREQVAL NUMCOLS 1 COUNT 5
   FREQVAL NUMCOLS 2 COUNT 10
   REPORT YES
```

**Example 12: Updating distribution statistics for a group of specified columns in a table**

The following control statement specifies that RUNSTATS is to update statistics for the columns EMPLEVEL, EMPGRADE, and EMPSALARY in table DSN8810.DEPT (in table space DSN8D81A.DSN8S81E). The statement uses the COLGROUP keyword to group these columns. RUNSTATS is to collect the cardinality of this column group and store the cardinality in the SYSCOLDIST catalog table.

```
RUNSTATS TABLESPACE DSN8D81A.DSN8S81E
   TABLE(DSN8810.DEPT)
   COLGROUP (EMPLEVEL,EMPGRAGE,EMPSALARY)
```

**Example 13: Updating distribution statistics for specific columns and retrieving the most frequently occurring values**

The following control statement specifies that RUNSTATS is to update statistics for the columns EMPLEVEL, EMPGRADE, and EMPSALARY in table DSN8810.DEPT. The FREQVAL and COUNT options indicate that RUNSTATS is to collect the 10 most frequently occurring values for each column. The values are to be stored in the SYSCOLDIST and SYSCOLDISTSTATS catalog tables.

```
RUNSTATS TABLESPACE DSN8D81A.DSN8S81E
   TABLE(DSN8810.DEPT)
   COLGROUP(EMPLEVEL,EMPGRAGE,EMPSALARY) FREQVAL COUNT 10
```

**Example 14: Updating distribution statistics for specific columns in a table and retrieving the least frequently occurring values**

The following control statement specifies that RUNSTATS is to update statistics for the columns EMPLEVEL, EMPGRADE, and EMPSALARY in table DSN8810.DEPT. The FREQVAL and COUNT options indicate that RUNSTATS is to collect the 15 least frequently occurring values for each column. The values are to be stored in the SYSCOLDIST and SYSCOLDISTSTATS catalog tables.

```
RUNSTATS TABLESPACE DSN8D81A.DSN8S81E
   TABLE(DSN8810.DEPT)
   COLGROUP(EMPLEVEL,EMPGRAGE,EMPSALARY) FREQVAL COUNT 15 LEAST
```

**Example 15: Updating distribution statistics for specific columns in a table space and retrieving the most and least frequently occurring values**

The following control statement specifies that RUNSTATS is to update statistics for the columns EMPLEVEL, EMPGRADE, and EMPSALARY in table DSN8810.DEPT. The FREQVAL and COUNT options indicate that RUNSTATS is to collect the 10
most frequently occurring values for each column and the 10 least frequently occurring values for each column. The values are to be stored in the SYSCOLDIST and SYSCOLDISTSTATS catalog tables.

```
RUNSTATS TABLESPACE DSN8D81A.DSN8S81E
  TABLE(DSN8810.DEPT)
  COLGROUP(EMPLEVEL, EMPGRADE, EMPSALARY) FREQVAL COUNT 10 BOTH
```

**Example 16: Updating statistics for an index and retrieving the most and least frequently occurring values**

The following control statement specifies that RUNSTATS is to collect the 10 most frequently occurring values and the 10 least frequently occurring values for the first key column of index ADMF001.IXMA0101. By default, the utility collects all the distinct values in all the key column combinations. A set of messages is sent to SYSPRINT and all collected statistics are updated in the catalog.

```
RUNSTATS INDEX(ADMF001.IXMA0101)
  FREQVAL NUMCOLS 1 COUNT 10 BOTH
  REPORT YES UPDATE ALL
```

**Example 17: Invalidating statements in the dynamic statement cache for a table space without generating report statistics.**

The following control statement specifies that RUNSTATS is to invalidate statements in the dynamic statement cache for table space DSN8D81A.DSN8S81E. However, RUNSTATS is not to collect or report statistics or update the catalog.

```
RUNSTATS TABLESPACE DSN8D81A.DSN8S81E
  REPORT NO
  UPDATE NONE
```

**Example 18: RUNSTATS HISTOGRAM job statement.**

The following control statement specifies that RUNSTATS is to gather histogram statistics for the specified key columns. Histogram statistics can only be collected on the prefix columns with the same order.

```
RUNSTATS TABLESPACE RVTDB01.RVTTS01
  INDEX ALL
  HISTOGRAM NUMCOLS 2  NUMQUANTILES 5
  SHRLEVEL(CHANGE)
  UPDATE ALL
  REPORT YES
```
Chapter 33. STOSPACE

The STOSPACE online utility updates Db2 catalog columns that indicate how much space is allocated for storage groups and related table spaces and indexes.

For user-defined spaces, STOSPACE does not record any statistics.

Output

The output from STOSPACE consists of updated values in the columns and tables in the following list. In each case, an amount of space is given in kilobytes (KB). If the value is too large to fit in the SPACE column, the SPACEF column is updated.

- SPACE in SYSIBM.SYSINDEXES shows the amount of space that is allocated to indexes. If the index is not defined using STOGROUP, or if STOSPACE has not been executed, the value is zero.
- SPACE in SYSIBM.SYSTABLESPACE shows the amount of space that is allocated to table spaces. If the table space is not defined using STOGROUP, or if STOSPACE has not been executed, the value is zero.
- SPACE in SYSIBM.SYSINDEXPART shows the amount of space that is allocated to index partitions. If the partition is not defined using STOGROUP, or if STOSPACE has not been executed, the value is zero.
- SPACE in SYSIBM.SYSTABLEPART shows the amount of space that is allocated to table partitions. If the partition is not defined using STOGROUP, or if STOSPACE has not been executed, the value is zero.
- SPACE in SYSIBM.SYSSTOGROUP shows the amount of space that is allocated to storage groups.
- STATSTIME in SYSIBM.SYSSTOGROUP shows the timestamp for the time at which STOSPACE was last executed.

Authorization required

To execute this utility, you must use a privilege set that includes one of the following authorities:

- STOSPACE privilege
- SYSCTRL or SYSADM authority

Execution phases of STOSPACE

The STOSPACE utility operates in these phases:

Phase  Description

UTILINIT  Performs initialization

STOSPACE  Gathers space information and updates catalog

UTILTERM  Performs cleanup
Syntax and options of the STOSPACE control statement

The STOSPACE utility control statement, with its multiple options, defines the function that the utility job performs.

You can create a control statement with the ISPF/PDF edit function. After creating it, save it in a sequential or partitioned data set. When you create the JCL for running the job, use the SYSIN DD statement to specify the name of the data set that contains the utility control statement.

Syntax diagram

![Syntax Diagram](image_url)

Option descriptions

**STOGROUP**
Identifies the storage groups that are to be processed.

\[(stogroup-name, \ldots)\]

Specifies the name of a storage group. You can use a list of from one to 255 storage group names. Separate items in the list by commas, and enclose them in parentheses.

* Indicates that all storage groups are to be processed.

Data sets that STOSPACE uses

The STOSPACE utility uses a number of data sets during its operation.

The following table lists the data sets that STOSPACE uses. The table lists the DD name that is used to identify the data set, a description of the data set, and an indication of whether it is required. Include statements in your JCL for each required data set and any optional data sets that you want to use.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSIN</td>
<td>Input data set that contains the utility control statement.</td>
<td>Yes</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>Output data set for messages.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

The following object is named in the utility control statement and does not require a DD statement in the JCL:

**Storage group**
Object that is to be reported.

Related concepts:
“Data sets that online utilities use” on page 16
**Concurrency and compatibility for STOSPACE**

The STOSPACE utility has certain concurrency and compatibility characteristics associated with it.

STOSPACE does not set a utility restrictive state on the target object.

STOSPACE can run concurrently with any utility on the same target object. However, because STOSPACE updates the catalog, concurrent STOSPACE utility jobs or other concurrent applications that update the catalog might cause timeouts and deadlocks.

You can use the STOSPACE utility on storage groups that have objects within temporary databases.

**How STOSPACE ensures availability of objects it STOSPACE requires**

For each specified storage group, STOSPACE looks at the `SYSTABLESPACE` and `SYSINDEXES` catalog tables to determine which objects belong to that storage group.

For each object, the amount of allocated space is determined from an appropriate VSAM catalog. Hence the table spaces and indexes do not need to be available to Db2 when STOSPACE is running; only the Db2 catalog and appropriate VSAM catalogs are required. However, to gain access to the VSAM catalog, the utility must have available to it the database definition (DBD) for the objects that are involved. This access requires that the appropriate database, table spaces, and index spaces not be in the stopped state.

**Obtaining statistical information with STOSPACE**

When Db2 storage groups are used in the creation of table spaces and indexes, Db2 defines the data sets for them. The STOSPACE utility permits a site to monitor the disk space that is allocated for the storage group.

**About this task**

The following table lists statistical information that the STOSPACE utility records and that is useful for making space allocation decisions.

<table>
<thead>
<tr>
<th>Catalog table</th>
<th>Column name</th>
<th>Column description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSTABLESPACE</td>
<td>SPACEF</td>
<td>Number of kilobytes of storage that are allocated to the table space</td>
</tr>
<tr>
<td>SYSTABLEPART</td>
<td>SPACEF</td>
<td>Number of kilobytes of storage that are allocated to the table space partition</td>
</tr>
<tr>
<td>SYSINDEXES</td>
<td>SPACEF</td>
<td>Number of kilobytes of storage that are allocated to the index</td>
</tr>
<tr>
<td>SYSINDEXEXPART</td>
<td>SPACEF</td>
<td>Number of kilobytes of storage that are allocated to the index partition</td>
</tr>
</tbody>
</table>
**Table 112. Db2 catalog data that STOSPACE collects (continued)**

<table>
<thead>
<tr>
<th>Catalog table</th>
<th>Column name</th>
<th>Column description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSTOGROUP</td>
<td>SPACEF</td>
<td>Number of kilobytes of storage that are allocated to the storage group</td>
</tr>
<tr>
<td>SYSTOGROUP</td>
<td>STATSTIME</td>
<td>Time when STOSPACE was last run on a particular storage group</td>
</tr>
</tbody>
</table>

STOSPACE does not accumulate information for more than one storage group. If a partitioned table space or index space has partitions in more than one storage group, the information in the catalog about that space comes from only the group for which STOSPACE was run.

When you run the STOSPACE utility, the SPACEF column of the catalog represents the high-allocated RBA of the VSAM linear data set. Use the value in the SPACEF column to project space requirements for table spaces, table space partitions, index spaces, and index space partitions over time. Use the output from the Access Method Services LISTCAT command to determine which table spaces and index spaces have allocated secondary extents. When you find these, increase the primary quantity value for the data set, and run the REORG utility.

**Procedure**

- For information about space utilization in the DSN8S11E table space in the DSN8D11A database:
  1. Run the STOSPACE utility
  2. Execute the following SQL statement:

     ```sql
     EXEC SQL
     SELECT SPACE
     FROM SYSIBM.SYSTABLESPACE
     WHERE NAME = 'DSN8S11E'
     AND DBNAME = 'DSN8D11A'
     ENDEXEC
     ```

     Alternatively, you can use TSO to look at data set and pack descriptions.

- To update SYSIBM.SYSTOGROUP for storage group DSN8G110, as well as SYSIBM.SYSTABLESPACE and SYSIBM.SYSINDEXES, for every table space and index that belongs to DSN8G110, use the following utility control statement:

  ```sql
  STOSPACE STOGROUP DSN8G110
  ```

**Analysis of the values in a SPACE or SPACEF column**

The value in a SPACE or SPACEF column represents total allocated space, not only the space that is allocated on the current list of volumes in the storage groups. If the value is too large to fit in the SPACE column, the SPACEF column is used.
You can delete volumes from a storage group even though space on those volumes is still allocated to Db2 table spaces or indexes. Deletion of a volume from a storage group prevents future allocations; it does not withdraw a current allocation.

**Termination or restart of STOSPACE**

You can terminate and restart the STOSPACE utility.

You can terminate a STOSPACE utility job with the **TERM UTILITY** command if you have submitted the job or have SYSOPR, SYSCTRL, or SYSADM authority.

You can restart a STOSPACE utility job, but it starts from the beginning again.

**Related tasks:**

“Restarting an online utility” on page 44

**Sample STOSPACE control statement**

Use the sample control statements as models for developing your own STOSPACE control statements.

**Example 1: Updating catalog SPACE columns for a particular storage group**

The following control statement specifies that the STOSPACE utility is to update the catalog SPACE or SPACEF columns for storage group DSN8G110 and any related table spaces and indexes.

```
//STEP1  EXEC DSNUPROC,UID='FUAUU330.STOSPCE',
    //  UTPROC='',
    //  SYSTEM='DSN'
//SYSIN DD *
STOSPACE STOGROUP DSN8G110
/*
```

**Example 2: Specifying a storage group name that contains spaces**

If the name of the storage group that you want STOSPACE to process contains spaces, enclose the entire storage group name in single quotation marks. Parentheses are optional. The following statements are correct ways to specify a storage group with the name THIS IS STOGROUP.1.ONE:

```
STOSPACE STOGROUP('THIS IS STOGROUP.1.ONE')
```

```
STOSPACE STOGROUP 'THIS IS STOGROUP.1.ONE'
```

**Example 3: Updating catalog SPACE columns for all storage groups**

The following control statement specifies that the STOSPACE utility is to update the catalog SPACE or SPACEF columns for all storage groups.

```
STOSPACE STOGROUP *
```
Example 4: Updating catalog SPACE columns for several storage groups

The following control statement specifies that the STOSPACE utility is to update the catalog SPACE or SPACEF columns for storage groups DSN8G110 and DSN8G81U.

STOSPACE STOGROUP(DSN8G810, DSN8G81U)
Chapter 34. TEMPLATE

The TEMPLATE online utility control statement lets you allocate data sets, without using JCL DD statements, during the processing of a LISTDEF list. The TEMPLATE control statement defines the data set naming convention. TEMPLATE control statements can also be written to contain allocation parameters that define data set size, location, and attributes.

Templates enable you to standardize data set names across the Db2 subsystem and to easily identify the data set type when you use variables in the data set name.

The TEMPLATE control statement uses the z/OS DYNALLOC macro (SVC 99) to perform data set allocation. Therefore, the facility is constrained by the limitations of this macro and by the subset of DYNALLOC that is supported by TEMPLATE. See z/OS MVS Programming: Authorized Assembler Services Guide for more details.

Output

The TEMPLATE control statement generates a dynamic allocation template with an assigned name for later reference.

Authorization required

No privileges are required to execute this control statement. When a TEMPLATE is referenced by a specific utility, privileges are checked at that time.

Execution phases of TEMPLATE

The TEMPLATE control statement executes entirely in the UTILINIT phase, which performs setup for the subsequent utility.

Syntax and options of the TEMPLATE control statement

The TEMPLATE utility control statement, with its multiple options, defines the function that the utility job performs.

You can create a control statement with the ISPF/PDF edit function. After creating it, save it in a sequential or partitioned data set. When you create the JCL for running the job, use the SYSIN DD statement to specify the name of the data set that contains the utility control statement.

Syntax diagram
name-expression:

Notes:
1. The entire name-expression represents one character string and cannot contain any blanks.

qualifier-expression:

Notes:
1. If you use substring notation, the entire DSN operand must be enclosed in single quotation marks. For example, the DSN operand 'P&PA(4,2).' uses substring notation, so it is enclosed in single quotation marks.
2. The &PA. variable cannot be used more than once.

common-options:
SUBSYS-spec

path-expression
Notes:

1. For LOAD, the default is PATHOPTS(ORDONLY).
2. For UNLOAD, the default is PATHOPTS(OCREAT,OWRONLY).

Option descriptions

**TEMPLATE** template-name

Defines a data set allocation template and assigns to the template a name, *template-name*, for subsequent reference on a Db2 utility control statement. The *template-name* can have up to eight alphanumeric characters and must begin with an alphabetic character.

*template-name* has the following additional restrictions:
- Cannot be UTPRINT or SORTLIB
- Cannot begin with SORTWK or SYS
- Cannot be a utility keyword name
The template-name is followed by keywords that control the allocation of tape and disk data sets. A single TEMPLATE statement cannot have both disk options and tape options. The UNIT keyword specifies a generic unit name that is defined on your system. This value is used to determine if a disk or tape data set is being allocated. All other keywords specified on the TEMPLATE control statement must be consistent with the specified unit type.

**DSN name-expression**

Specifies the template for the z/OS data set name. You can specify the data set name, name-expression, by using symbolic variables, non-variable alphanumeric, or national characters, or any combination of these characters. The resulting name must adhere to the z/OS data set naming rules, including those rules about name length, valid characters, name structure and qualifier length. You must specify a DSN expression that is unique for each data set allocated by the utility and to each invocation of the utility.

Templates for FlashCopy image copies should specify only DSN name-expression. Db2 does not use any other options in a TEMPLATE control statement for FlashCopy image copies.

Data set names consists of a series of qualifiers, qualifier-expression, that are separated by a period (.) and an optional parenthetical expression. No imbedded blanks are allowed. A partitioned data set (PDS) cannot be defined by TEMPLATE for use as an input data set.

If the DSN name operand contains any special characters, it must be enclosed in single quotation marks. For example, in the following TEMPLATE statement, the DSN operand contains the parentheses special character, so the entire operand is enclosed in single quotation marks:

```
TEMPLATE X DSN 'A.GDG.VERSION(+1)'
```

Parentheses around the DSN name operand are optional. They are used in the following DSN specification:

```
DSN(&DB..&TS..D&DATE.)
```

**character-expression**

Specifies the data set name or part of the data set name by using non-variable alphanumeric or national characters.

**parenthetical-expression**

Specifies part of the data set name by using non-variable alphanumeric or national characters that are enclosed in parentheses. For example, the expressions Q1.Q2.Q3(member) and Q1.Q2.Q3(+1) use valid parenthetical expressions. No variable substitution is performed within the parenthetical expression.

**&variable.**

Specifies the data set name or part of the data set name by using symbolic variables. See the following tables for a list of variables that can be used.

Each symbolic variable is substituted with its related value at execution time to form a specific data set name. When used in a DSN expression, substitution variables begin with an ampersand sign (&) and end with a period (.), as in the following example:

```
DSN &DB..&TS..&D&DATE..COPY&ICTYPE.
```

Using numeric variables alone generates an invalid data set qualifier for all numeric-type variables (all date or time-type variables, and others, such as
&SEQ. or &PART.}. These variables must be preceded by character constants to form valid DSN qualifiers. The following examples are valid specifications:

P&PART.
D&DATE.

Some substitution variables are invalid if you use TEMPLATE with an incompatible utility. For example, ICTYPE is not meaningful if the TEMPLATE statement is used with LOAD SYSDISC. Other variables assume default values when their values are not known. For example, &PART becomes 00000 for non-partitioned objects.

You can also use substring notation for data set name variables. This notation can help you keep the data set name from exceeding the 44 character maximum. If you use substring notation, the entire DSN operand must be enclosed in single quotation marks. To specify a substring, use the form &variable(start). or &variable(start,length).

**start**

Specifies the substring's starting byte location within the current variable base value at the time of execution. **start** must be an integer from 1 to 128.

**length**

Specifies the length of the substring. If you specify **start** but do not specify **length**, by default, is the number of characters from the **start** character to the last character of the variable value at the time of execution. For example, given a five-digit base value, &PART(4). specifies the fourth and fifth digits of the value. **length** must be an integer that does not cause the substring to extend beyond the end of the base value.

For UNLOAD on a partitioned table space, if you use substring notation for the partition variable (&PART. or &PA.) in the DSN argument, the data set name might not be unique for all partitions, so Db2 cannot do parallel UNLOAD operations for the partitions. Therefore, Db2 sets the value of &PA to '00000', and uses a single UNLDDN data set for all partitions. This action might cause duplicate data set errors on subsequent UNLOAD jobs for other partitions of the same table space.

The following table contains a list of JOB variables and their descriptions.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;JOBNAME. or &amp;JO.</td>
<td>The z/OS job name.</td>
</tr>
<tr>
<td>&amp;STEPNAME. or &amp;ST.</td>
<td>The z/OS step name. This variable might be needed if data set names from two different job steps conflict.</td>
</tr>
<tr>
<td>&amp;USERID. or &amp;US.</td>
<td>The user ID of the person that is running the utility. The value is 1 to 8 characters long.</td>
</tr>
<tr>
<td>&amp;UTILID. or &amp;UT.</td>
<td>The utility ID truncated to eight characters and checked for invalid DSN characters.</td>
</tr>
<tr>
<td>&amp;SSID. or &amp;SS.</td>
<td>Subsystem ID (non-data sharing) or group attachment name (data sharing).</td>
</tr>
</tbody>
</table>

The following table contains a list of UTILITY variables and their descriptions.
### Table 114. UTILITY variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;ICTYPE. or &amp;IC.</td>
<td>Single-character image copy type. This variable is valid only for image copy templates. The substitution is governed by whether a full image copy (F), an incremental image copy (I), or a CHANGELIMIT image copy (C) is specified by the user.</td>
</tr>
<tr>
<td>&amp;UTILNAME. or &amp;UN.</td>
<td>Special values are assigned to some utilities: CHECKD for CHECK DATA, CHECKI for CHECK INDEX, CHECKL for CHECK LOB, REORGI for REORG INDEX, and REORGT for REORG TABLESPACE. Utility names that are longer than eight characters are truncated to eight characters.</td>
</tr>
<tr>
<td>&amp;SEQ. or &amp;SQ.</td>
<td>Sequence number of the LISTDEF list item being processed.</td>
</tr>
<tr>
<td>&amp;LOCREM. or &amp;LR.</td>
<td>Indicator of whether &amp;ddname is for the local site (COPYDDN) or the recovery site (RECOVERYDDN). Single character L is used when the utility defines a COPYDDN &amp;ddname. The single character R is used when the utility defines a RECOVERYDDN &amp;ddname. You can replicate the SYSCOPY ICBACKUP column information by using both the &amp;LOCREM. and &amp;PRIBAC. variables. This variable is valid only for image copy templates.</td>
</tr>
<tr>
<td>&amp;PRIBAC. or &amp;PB.</td>
<td>Indicator of whether &amp;ddname is for the primary (ddname1) or backup (ddname2) copy data set. Single character P is used when the utility defines a ddname1. The single character B is used when the utility defines a ddname2. You can replicate the SYSCOPY ICBACKUP column information by using both the &amp;LOCREM. and &amp;PRIBAC. variables. This variable is valid only for image copy templates.</td>
</tr>
</tbody>
</table>

The following table contains a list of OBJECT variables and their descriptions.

### Table 115. OBJECT variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;LIST. or &amp;LI.</td>
<td>The name of the list that is defined by using the LISTDEF control statement and that is referenced on the same control statement as this TEMPLATE. This variable is used with COPY FILTERDDN templates. All objects in the list are copied to one data set, which makes &amp;TS. and &amp;IS. meaningless.</td>
</tr>
<tr>
<td>&amp;DB.</td>
<td>Database name.</td>
</tr>
<tr>
<td>&amp;TS.</td>
<td>Table space name.</td>
</tr>
<tr>
<td>&amp;IS.</td>
<td>Index space name.</td>
</tr>
<tr>
<td>&amp;SN.</td>
<td>Space name (table space or index space).</td>
</tr>
<tr>
<td>&amp;PART. or &amp;PA.</td>
<td>Five-digit partition number, padded with leading zeros.</td>
</tr>
<tr>
<td>&amp;DSNUM</td>
<td>Five-digit partition number for partitioned objects, or five-digit piece number for linear objects, padded with leading zeroes.</td>
</tr>
</tbody>
</table>
Table 115. OBJECT variables (continued)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td></td>
</tr>
<tr>
<td>1. When you specify the &amp;TS., &amp;IS., or &amp;SN. variables in a template that is used by an UNLOAD statement with BLOBF, CLOBF, or DBCLOBF, Db2 substitutes the name of the table space that stores the LOB column value, not the base table space name. This substitution enables Db2 to generate unique data set names for each LOB column with partitioned table spaces.</td>
<td></td>
</tr>
<tr>
<td>2. Use the &amp;PA. variable when processing LISTDEF lists with the PARTLEVEL keyword or data-partitioned secondary indexes. Otherwise, Db2 could generate duplicate data set names.</td>
<td></td>
</tr>
<tr>
<td>3. Templates for FlashCopy image copies can contain either &amp;PART or &amp;DSNUM. If you are copying both partitioned and linear objects, use &amp;DSNUM.</td>
<td></td>
</tr>
<tr>
<td>4. FlashCopy image copies are always made at the partition or piece level. This behavior occurs even if you request FlashCopy image copies with DSNUM ALL on your COPY utility statement or include them in a LISTDEF list without the PARTLEVEL keyword. Therefore, for templates with the &amp;PART, &amp;PA., or &amp;DSNUM. variables for FlashCopy image copies, the number and names of data sets that result might differ from the result when the same templates are used by other utilities. The result might also differ from the data set names that are displayed when you preview the templates. (For information about previewing templates, see &quot;Key TEMPLATE operations&quot; on page 810.)</td>
<td></td>
</tr>
</tbody>
</table>

The following table contains a list of DATE and TIME variables and their descriptions.

Table 116. DATE and TIME variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;DATE. or &amp;DT.</td>
<td>YYYYDDD</td>
</tr>
<tr>
<td>&amp;TIME. or &amp;TI.</td>
<td>HHMMSS</td>
</tr>
<tr>
<td>&amp;JDATE. or &amp;JU.</td>
<td>YYYYDDD</td>
</tr>
<tr>
<td>&amp;YEAR. or &amp;YE.</td>
<td>YYYY portion of &amp;DATE.</td>
</tr>
<tr>
<td>&amp;MONTH. or &amp;MO.</td>
<td>MM</td>
</tr>
<tr>
<td>&amp;DAY. or &amp;DA.</td>
<td>DD</td>
</tr>
<tr>
<td>&amp;JDAY. or &amp;JD.</td>
<td>DDD portion of &amp;DATE.</td>
</tr>
<tr>
<td>&amp;HOUR. or &amp;HO.</td>
<td>HH portion of &amp;TIME.</td>
</tr>
<tr>
<td>&amp;MINUTE. or &amp;MI.</td>
<td>MM portion of &amp;TIME.</td>
</tr>
<tr>
<td>&amp;SECOND. or &amp;SC.</td>
<td>SS portion of &amp;TIME.</td>
</tr>
<tr>
<td>&amp;UNIQ. or &amp;UQ.</td>
<td>Unique eight characters that Db2 derives from the system clock at the time of allocation. This set of characters begins with an alphabetical character and is followed by seven alphabetical or numeric characters.</td>
</tr>
</tbody>
</table>

**Note:** Date and time values are set with the STCK instruction. The value is in local time or Coordinated Universal Time (UTC) depending on the TIME option or TEMPLATE_TIME subsystem parameter. Except for the &UNIQ. and &UQ. variables, DATE and TIME values are captured in the UTILINIT phase of each utility and remain constant until the utility terminates. &UNIQ. and &UQ. are assigned a unique value for each allocation.

**SUBSYS name**

Specifies the MVS BATCHPIPE SUBSYSTEM name. The SUBSYS operand must be a valid BATCHPIPE SUBSYSTEM name and must not exceed eight characters in length. When SUBSYS is specified, LRECL and RECFM are
required. When SUBSYS is specified, TEMPLATE keywords that are not compatible with SUBSYS (such as UNIT) are ignored.

**Restriction:** When using BATCHPIPES, TEMPLATE with the SUBSYS keyword, the utility cannot be restarted and the LOAD DISCARDDN keyword is not supported.

**LRECL int**

Specifies the record length of the MVS BATCHPIPES SUBSYSTEM file or z/OS UNIX file. You must specify LRECL if you specify SUBSYS.

LRECL does not have a default value except in the following situation: If you specify TEMPLATE PATH and accept the default value FILEDATA RECORD, the default value for LRECL is 32756.

**RECFM**

Specifies the record format of the MVS BATCHPIPES SUBSYSTEM file or z/OS UNIX file. You must specify RECFM if you specify SUBSYS.

Valid values for RECFM are F, FB, V, or VB.

RECFM does not have a default value except in the following situation: If you specify TEMPLATE PATH and accept the default value FILEDATA RECORD, the default value for RECFM is VB.

**UNIT**

Specifies the device-number, device-type (generic), or group-name for the data set. All other TEMPLATE keywords are validated based on the specified type of unit (disk or tape).

The default value is **SYSALLDA**.

**MODELDCB dsname**

Specifies the name of the data set on which the template is based. DCB information is read from this model data set.

**BUFNO integer**

Specifies the number of BSAM buffers. The specified value must be in the range 0 - 99.

The default value is **30**.

**DATACLAS name**

Specifies the SMS data class. The name value must be a valid SMS data class and must not exceed eight characters in length.

The data set is cataloged if DATACLAS is specified. If this option is omitted, no DATACLAS is specified to SMS.

**MGMTCLAS name**

Specifies the SMS management class. The name value must be a valid SMS management class and must not exceed eight characters in length.

The data set is cataloged if MGMTCLAS is specified. If this option is omitted, no MGMTCLAS is specified to SMS.

**STORCLAS name**

Specifies the SMS storage class. The name value must be a valid SMS storage class and must not exceed eight characters in length.

The data set is cataloged if STORCLAS is specified. If this option is omitted, no STORCLAS is specified to SMS.
RETPD integer
Specifies the retention period in days for the data set. The integer value must be in the range from 0 to 9999.

If DATACLAS, MGMTCLAS, or STORCLAS is specified, the class definition might control the retention. RETPD cannot be specified with EXPDL.

EXPDL 'date'
Specifies the expiration date for the data set, in the form YYYYDDD, where YYYY is the four-digit year, and DDD is the three-digit Julian day. The 'date' value must be enclosed by single quotation marks.

If DATACLAS, MGMTCLAS, or STORCLAS is specified, the class definition might control the retention. EXPDL cannot be specified with RETPD.

VOLUMES (vol1,vol2,...)
Specifies a list of volume serial numbers for this allocation. If the data set is not cataloged the list is truncated, if necessary, when it is stored in SYSIBM.SYSCOPY. The specified number of volumes cannot exceed the specified or default value of VOLCNT.

The first volume must contain enough space for the primary space allocation.

If an individual volume serial-number contains leading zeros, it must be enclosed in single quotation marks.

VOLCNT (integer)
Specifies the maximum number of volumes that an output data set might require. The specified value must be between 0 and 255.

The default value for tape templates is 95. For disk templates, the utility does not set a default value. Operating system defaults apply.

UNCNT integer
Specifies the number of devices that are to be allocated. The specified value must be in the range 0 - 59.

If UNIT specifies a specific device number, the value of UNCNT must either be 1 or be omitted.

GDGLIMIT (integer)
Specifies the number of entries that are to be created in a GDG base if a GDG DSN is specified and the base does not already exist. If a GDG base does not already exist and you do not want to define one, specify a GDGLIMIT of zero (0).

The default value is 99. The minimum value is 0. The maximum value depends on the z/OS release on which the subsystem is being executed. For z/OS V2R1 or earlier, the maximum value is 255. For z/OS V2R2 or later, the maximum value is 999. Before using values that are greater than 255, check with your database administrator or system programmer to ensure that GDGEXTENDED(YES) is specified in SYS1.PARMLIB(IGGCATxx). Otherwise, values that are greater than 255 will fail during allocation.

DISP (status, normal-termination, abnormal-termination)
Specifies the data set disposition by using three positional parameters: status, normal-termination, and abnormal-termination. All three parameters must be specified.

status
Standard z/OS values are allowed: NEW, OLD, SHR, MOD.
**normal-termination**

Standard z/OS values are allowed: DELETE, KEEP, CATLG, UNCATLG.

**abnormal-termination**

Standard z/OS values are allowed: DELETE, KEEP, CATLG, UNCATLG.

Default values for DISP vary, depending on the utility and the data set that is being allocated. Defaults for restarted utilities also differ from default values for new utility executions. Default values are shown in the following tables.

The following table shows the data dispositions for dynamically allocated data sets for new utility executions.

**Note:** It is possible that output from utilities that use piped data would not be dynamically allocated for new utility executions.

<table>
<thead>
<tr>
<th>ddname</th>
<th>CHECK DATA</th>
<th>CHECK INDEX or CHECK LOB</th>
<th>COPY</th>
<th>COPY-TOCOPY</th>
<th>LOAD</th>
<th>MERGE-COPY</th>
<th>REBUILD INDEX</th>
<th>REORG INDEX</th>
<th>REORG TABLE-SPACE</th>
<th>UNLOAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSREC</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>OLD KEEP KEEP</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>NEW CATLG</td>
<td>NEW CATLG</td>
</tr>
<tr>
<td>SYSDISC</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>NEW CATLG CATLG</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>NEW CATLG</td>
<td>Ignored</td>
</tr>
<tr>
<td>SYSPUNCH</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>NEW CATLG</td>
<td>NEW CATLG</td>
<td></td>
</tr>
<tr>
<td>SYSCOPY</td>
<td>Ignored</td>
<td>Ignored</td>
<td>NEW CATLG CATLG</td>
<td>Ignored</td>
<td>NEW CATLG CATLG</td>
<td>NEW CATLG CATLG</td>
<td>Ignored</td>
<td>NEW CATLG</td>
<td>Ignored</td>
<td></td>
</tr>
<tr>
<td>SYSCOPY2</td>
<td>Ignored</td>
<td>Ignored</td>
<td>NEW CATLG CATLG</td>
<td>Ignored</td>
<td>NEW CATLG CATLG</td>
<td>NEW CATLG CATLG</td>
<td>Ignored</td>
<td>NEW CATLG</td>
<td>Ignored</td>
<td></td>
</tr>
<tr>
<td>SYSRCPY1</td>
<td>Ignored</td>
<td>Ignored</td>
<td>NEW CATLG CATLG</td>
<td>Ignored</td>
<td>NEW CATLG CATLG</td>
<td>NEW CATLG CATLG</td>
<td>Ignored</td>
<td>NEW CATLG</td>
<td>Ignored</td>
<td></td>
</tr>
<tr>
<td>SYSRCPY2</td>
<td>Ignored</td>
<td>Ignored</td>
<td>NEW CATLG CATLG</td>
<td>Ignored</td>
<td>NEW CATLG CATLG</td>
<td>NEW CATLG CATLG</td>
<td>Ignored</td>
<td>NEW CATLG</td>
<td>Ignored</td>
<td></td>
</tr>
<tr>
<td>SYSUT1</td>
<td>NEW DELETE CATLG</td>
<td>NEW DELETE CATLG</td>
<td>Ignored</td>
<td>Ignored</td>
<td>NEW DELETE CATLG</td>
<td>NEW DELETE CATLG</td>
<td>NEW DELETE CATLG</td>
<td>NEW DELETE CATLG</td>
<td>Ignored</td>
<td></td>
</tr>
<tr>
<td>SORTOUT</td>
<td>NEW DELETE CATLG</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>NEW DELETE CATLG</td>
<td>NEW DELETE CATLG</td>
<td>NEW DELETE CATLG</td>
<td>NEW DELETE CATLG</td>
<td>Ignored</td>
<td></td>
</tr>
<tr>
<td>SYSMAP</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>NEW CATLG CATLG</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td></td>
</tr>
<tr>
<td>SYSERR</td>
<td>NEW CATLG CATLG</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>NEW CATLG CATLG</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td></td>
</tr>
<tr>
<td>FILTERDDS</td>
<td>Ignored</td>
<td>Ignored</td>
<td>NEW DELETE DELETE</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td></td>
</tr>
</tbody>
</table>

The following table shows data dispositions for dynamically allocated data sets on RESTART.
<table>
<thead>
<tr>
<th>ddname</th>
<th>CHECK DATA</th>
<th>CHECK INDEX or CHECK LOB</th>
<th>COPY</th>
<th>COPY-TOCOPY</th>
<th>LOAD</th>
<th>MERGE-COPY</th>
<th>REBUILD INDEX</th>
<th>REORG INDEX</th>
<th>REORG TABLE-SPACE</th>
<th>UNLOAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSREC</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>OLD KEEP KEEP</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>MOD CATLG CATLG</td>
<td>MOD CATLG CATLG</td>
</tr>
<tr>
<td>SYSDISC</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>MOD CATLG CATLG</td>
<td>Ignored</td>
<td>Ignored</td>
<td>MOD CATLG CATLG</td>
<td>MOD CATLG</td>
<td>Ignored</td>
<td></td>
</tr>
<tr>
<td>SYSPUNCH</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>MOD CATLG CATLG</td>
<td>MOD CATLG</td>
<td>Ignored</td>
<td></td>
</tr>
<tr>
<td>SYSCOPY</td>
<td>Ignored</td>
<td>MOD CATLG CATLG</td>
<td>Ignored</td>
<td>MOD CATLG CATLG</td>
<td>Ignored</td>
<td>MOD CATLG CATLG</td>
<td>MOD CATLG CATLG</td>
<td>MOD CATLG</td>
<td>Ignored</td>
<td></td>
</tr>
<tr>
<td>SYSCOPY2</td>
<td>Ignored</td>
<td>MOD CATLG CATLG</td>
<td>Ignored</td>
<td>MOD CATLG CATLG</td>
<td>Ignored</td>
<td>MOD CATLG CATLG</td>
<td>MOD CATLG CATLG</td>
<td>MOD CATLG</td>
<td>Ignored</td>
<td></td>
</tr>
<tr>
<td>SYSRCPY1</td>
<td>Ignored</td>
<td>MOD CATLG CATLG</td>
<td>Ignored</td>
<td>MOD CATLG CATLG</td>
<td>Ignored</td>
<td>MOD CATLG CATLG</td>
<td>MOD CATLG CATLG</td>
<td>MOD CATLG</td>
<td>Ignored</td>
<td></td>
</tr>
<tr>
<td>SYSRCPY2</td>
<td>Ignored</td>
<td>MOD CATLG CATLG</td>
<td>Ignored</td>
<td>MOD CATLG CATLG</td>
<td>Ignored</td>
<td>MOD CATLG CATLG</td>
<td>MOD CATLG CATLG</td>
<td>MOD CATLG</td>
<td>Ignored</td>
<td></td>
</tr>
<tr>
<td>SYSUT1</td>
<td>MOD DELETE</td>
<td>MOD DELETE CATLG</td>
<td>Ignored</td>
<td>MOD DELETE CATLG</td>
<td>Ignored</td>
<td>MOD DELETE CATLG</td>
<td>MOD DELETE CATLG</td>
<td>MOD DELETE CATLG</td>
<td>Ignored</td>
<td></td>
</tr>
<tr>
<td>SORTOUT</td>
<td>MOD DELETE</td>
<td>CATLG</td>
<td>Ignored</td>
<td>MOD DELETE CATLG</td>
<td>Ignored</td>
<td>MOD DELETE CATLG</td>
<td>MOD DELETE CATLG</td>
<td>MOD DELETE CATLG</td>
<td>Ignored</td>
<td></td>
</tr>
<tr>
<td>SYSMAP</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>MOD CATLG CATLG</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>MOD CATLG CATLG</td>
<td>Ignored</td>
<td></td>
</tr>
<tr>
<td>SYSERR</td>
<td>MOD CATLG CATLG</td>
<td></td>
<td>Ignored</td>
<td>MOD CATLG CATLG</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>MOD CATLG CATLG</td>
<td>Ignored</td>
<td></td>
</tr>
<tr>
<td>FILTERDDS</td>
<td>Ignored</td>
<td>NEW DELETE DELETE</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Ignored</td>
</tr>
</tbody>
</table>

**LIMIT**

Specifies template switching.

- **n** Specifies the maximum primary allocation quantity that is permitted using this TEMPLATE.

**CYL**

Cylinders

**GB**

Gigabytes

**MB**

Megabytes

**new_template**

Specifies a character string that specifies the name of a TEMPLATE to use if the size limit is exceeded.

Db2 supports the LIMIT keyword only on TEMPLATE control statements reference by COPYDDN or RECOVERYDDN keywords on the following utilities:

- **COPY FULL YES|NO**
- **COPY CONCURRENT**
- **COPYTOCOPY**
Restriction:
- You cannot switch to a DD card.
- The template control statement that LIMIT references must exist in SYSIN or SYSTEMPL and it cannot refer to itself.
- Switching can only be performed a single time per allocation. Multiple switching cannot take place.
- The utility PREVIEW function ignores the LIMIT keyword, only the original TEMPLATE control statement is previewed. The LIMIT keyword is ignored for new templates.

TIME
Specifies time used in expansion of date and time DSN variables. The default TIME value is determined by the TEMPLATE_TIME subsystem parameter.

Tip: Set all Db2 data sharing members to the same value.

LOCAL
Use local time at the Db2 server in the expansion of date and time in DSN variables.

UTC
Use Coordinated Universal Time (UTC) in the expansion of date and time in DSN variables.

SPACE (primary, secondary)
Specifies the z/OS disk space allocation parameters in the range 1 - 1677215. If you specify (primary,secondary) value, these values are used instead of the Db2-calculated values. When specifying primary and secondary quantities, you must either specify both values or omit both values.

Use the MAXPRIME option to set an upper limit on the primary quantity.

CYL
Specifies that allocation quantities, if present, are to be expressed in cylinders and that allocation is to occur in cylinders. If SPACE CYL is specified, without (primary, secondary), the Db2-calculated quantities are allocated in cylinders by using 3390 device capacities for byte conversion. If TRK and MB are omitted, CYL is the default.

TRK
Specifies that allocation quantities, if present, are to be expressed in tracks and that allocation is to occur in tracks. If SPACE TRK is specified, without (primary, secondary), the Db2-calculated quantities are allocated in tracks by using 3390 device capacities for byte conversion.

MB
Specifies that allocation quantities, if present, are to be expressed in megabytes, and that allocation is to occur in records. One megabyte is 1,048,576 bytes. If SPACE MB is specified, the (primary,secondary) quantities that are specified, or the Db2-calculated quantities, might be allocated in tracks or cylinders. Data sets with a primary or secondary allocation quantity greater than 20 MB are allocated in cylinders. Smaller data sets are allocated in tracks. The 3390 device capacities are used for TRK or CYL conversion.
**PCTPRIME** *integer*

Specifies the percentage of the estimated required space that is to be obtained as the primary quantity.

The default value is **100**.

Use the MAXPRIME option to set the upper limit of this value for large objects.

**MAXPRIME** *integer*

Specifies the maximum allowable primary space allocation, expressed in cylinders (CYL). This value constrains the primary space value and the PCTPRIME calculation, as well as the size of each secondary allocation.

**NBRSECDN** *integer*

Specifies the division of secondary space allocations. After the primary space is allocated, an amount of space equal to the estimated required space is divided into the specified number of secondary allocations. Individual utilities might request larger secondary extents to compensate for localized uncertainty in the space estimations.

The integer value must be in the range 1 - 10. The default value is **10**.

**DIR** *integer*

Specifies the number of 256-byte records that are to be allocated for the directory of a new partitioned data set. You must specify this operand if you are allocating a new partitioned data set.

If the template is being used in a UNLOAD statement with BLOBF, CLOBF, or DBCLOBF and you specify a DSNTYPE of LIBRARY or PDS, but do not specify DIR, Db2 calculates the number of 256-byte records to allocate by dividing the estimated number of records by 20.

**DSNTYPE**

Specifies the type of data set to be allocated.

**LIBRARY**

Specifies that a partitioned data set extended (PDSE) is to be allocated.

**PDS**

Specifies that a partitioned data set (PDS) is to be allocated.

**HFS**

Specifies that a hierarchical file system (HFS) file is to be allocated.

**NULL**

Specifies a null file. Use this value for a template with UNLOAD CLOBF, BLOBF, or DBCLOBF to unload a null LOB value. In this case, the unload data set contains a null file name.

**BASIC**

Specifies a basic format data set. No more than 65535 tracks can be allocated.

**LARGE**

Specifies a large format data set. Greater than 65535 tracks can be allocated.

**EXTREQ**

Specifies an extended format data set is required.

**EXTPREF**

Specifies an extended format data set is preferred.
If you omit DSNTYPE, the type of data set is determined by other data set attributes, the data class for the data set, or an installation default.

EATTR
Specifies that the data set can support extended attributes.

STACK
Specifies whether output data sets are to be stacked contiguously on the same tape volumes.

NO  Specifies that output data sets are not to be stacked contiguously on tape.

YES  Specifies that similar output data sets are to be stacked as successive files on one logical tape volume, where a logical tape volume can consist of a multi-volume aggregate. Within one utility execution, output data sets are stacked on a logical tape volume of the same usage type. For example, local primary image copies are stacked separately from local backup image copies.

Related information:
“Guidelines for templates and tape data sets” on page 813

TRTCH
Specifies the track recording technique for magnetic tape drives that have improved data recording capability.

NONE  Specifies that the TRTCH specification is to be eliminated from dynamic allocation.

COMP  Specifies that data is to be written in compacted format.

NOCOMP  Specifies that data is to be written in standard format.

PATH
Specifies a z/OS UNIX file path name, which can be the name of a Unix System Services (USS) pipe, an HFS file, or a zFS file.

Restrictions:
• If you specify PATH for a template, the utility that uses that template cannot be restarted.
• You can use a template with PATH only for input data sets for the LOAD utility (as indicated by the INDDN option) and for output data sets for the UNLOAD utility (as indicated by the UNLDDN option). You cannot use these templates for DISCARDDN data sets for the LOAD and REORG utilities.

When you specify PATH, adhere to the following requirements:
• Specify the path name in SBCS EBCDIC format.
• Do not specify a path name that is longer than 255 bytes.
• If the path name contains blanks, enclose it in single quotes.
• If you specify PATH and do not specify FILEDATA(RECORD), specify values for LRECL and RECFM.

FILEDATA
Specifies the content type of the z/OS UNIX file that is specified for the
PATH option. Valid values are TEXT, BINARY, and RECORD. RECORD indicates that the file contains both binary and text data.

The default value is RECORD.

PATHOPTS
Specifies the access and status for the z/OS UNIX file that is specified for the PATH option.

You can specify one or more of the following z/OS options for PATHOPTS:
- ORDONLY
- OCREAT
- OWRONLY
- ONONBLOCK

For information about these options, see PATHOPTS Parameter (MVS JCL Reference)

The default for LOAD is ORDONLY. The default for UNLOAD is OCREAT, OWRONLY.

PATHMODE
Specifies the file mode of the HFS file that is specified in the PATH option.

You can specify one or more of the following z/OS options for PATHMODE:
- SIRUSR
- SIWUSR
- SIXUSR
- SIRWXU
- SIRGRP
- SIWGRP
- SIXGRP
- SIRWXG
- SIROTH
- SIWOTH
- SIXOTH
- SIRWXO

For information about these options, see PATHMODE Parameter (MVS JCL Reference)

The default value is SIRUSR.

PATHDISP
Specifies the disposition of the z/OS UNIX file that is specified for the PATH option.

You must specify two parameters for the PATHDISP:
- The first parameter specifies whether the file is to be kept or deleted when the job ends normally.
- The second parameter specifies whether the file is to be kept or deleted when the job ends abnormally.

The valid values for each parameter are KEEP or DELETE.

The default value is KEEP, KEEP.

Related reference:
Before running **TEMPLATE**

Some Db2 utilities produce data sets during execution. These data sets are referenced in utility control statements by a set of DD name keywords and are specified in the corresponding JCL. Alternatively, you can use the TEMPLATE utility control statement to dynamically allocate utility data sets.

Options of the TEMPLATE utility allow you to specify the following information:

- The data set naming convention
- DFSMS parameters
- Disk or tape allocation parameters

You can specify a template in the SYSIN data set, immediately preceding the utility control statement that references it, or in one or more TEMPLATE libraries.

A TEMPLATE library is a data set that contains only TEMPLATE utility control statements. You can specify a TEMPLATE data set DD name by using the TEMPLATEDD option of the OPTIONS utility control statement. This specification applies to all subsequent utility control statements until the end of input or until Db2 encounters a new OPTIONS TEMPLATEDD(ddname) specification.

Any template that is defined within SYSIN overrides another template definition of the same name in a TEMPLATE data set.

TEMPLATE utility control statements enable you to standardize data set allocation and the utility control statements that reference those data sets, which reduces the need to customize and alter utility job streams.

**Concurrency and compatibility for TEMPLATE**

The TEMPLATE utility has certain concurrency and compatibility characteristics associated with it.

TEMPLATE is a control statement that is used to set up an environment for another utility to follow. The template is stored until it is referenced by a specific utility. The list is expanded when it is referenced by another utility. At that time, the concurrency and compatibility restrictions of that utility apply, and the catalog tables that are necessary to expand the list must be available for read-only access.

**Key TEMPLATE operations**

A TEMPLATE control statement performs a setup operation in preparation for use by another utility. Information is saved under the template name for the duration of the job step. For most utilities, you can reference the information by substituting the template name for the DD name.
If a DD name and a TEMPLATE name conflict, the DD statement is used for allocation, and the TEMPLATE is ignored. Minimally, a TEMPLATE statement consists of a name (similar to a DD name) and a data set naming convention. If nothing else is specified, Db2 calculates the required data set size and uses default data set attributes that are appropriate to the data set that is being created. Db2 then allocates a disk data set with these defaults.

The required TEMPLATE statement might look something like the following TEMPLATE statement:

```
TEMPLATE tmp1  DSN(DB2..&TS..D&JDATE..COPY&ICTYPE..&LOCREM..&PRIBAC.)
                  VOLUMES(vol1,vol2,vol3)
LISTDEF payroll INCLUDE TABLESPACE PAYROLL.*
                  INCLUDE INDEXSPACE PAYROLL.*IX
                  EXCLUDE TABLESPACE PAYROLL.TEMP*
                  EXCLUDE INDEXSPACE PAYROLL.TMPIX*
COPY LIST payroll  COPYDDN(tmp1,tmp1) RECOVERYDDN(tmp1,tmp1)
```

Database administrators can check utility control statements without executing them by using the PREVIEW function. In PREVIEW mode, Db2 expands all TEMPLATE data set names in the SYSIN DD, in addition to any data set name from the TEMPLATE DD that are referenced on a utility control statement. Db2 then prints the information to the SYSPRINT data set and halts execution. You can specify PREVIEW in one of two ways, either as a JCL PARM or on the OPTIONS PREVIEW utility control statement.

### Choosing data set names

The data set naming convention that is specified on the DSN option of each TEMPLATE statement must be appropriate for the data set that is being created. The data set naming convention must also be coordinated with the other templates and DD statements in the same job step.

#### About this task

The data set name must be both unique and meaningful. Db2 does not check that the data set names are unique until the execution of the utility that references the template. Ensure that the data set names are unique when you define the data set naming convention on the TEMPLATE control statement.

#### Procedure

To choose a data set name, apply the following guidelines:

- Use a combination of static characters, national characters, and the provided variable names to form valid z/OS data set qualifiers. Normal z/OS rules apply. Variables that produce numeric values must be preceded by either a static character or a character variable. All qualifiers must start with an alphabetic character. The qualifiers must consist of a maximum of eight characters and a maximum of 44 characters for the entire data set name. To help comply with this 44 character limit, you can use variable substring notation.
- Use the two-character form of the DSN variables to save space.
- Use two consecutive periods following all variables that precede the last qualifier (one to terminate the variable, followed by a second static period to separate the qualifiers), as in the following example:

  &DB..&TS.

- Use &DB. and &TS. to relate the data set to a database object.
• Use &PART. when executing PARTLEVEL lists. Precede the variable with a static character or a character variable to form a valid qualifier.
• Use &JO. and &ST. to eliminate conflicts with other jobs or job steps.
• Use &SS., &US., &UT., and &UN. if you have a need to know the subsystem, member, user, utility ID, or name of the utility that produced the data set.
• Use &DATE. and &TIME. or the shorter substring variations to guarantee uniqueness. Precede the variable with a static character or a character variable to form a valid qualifier.
• Use &IC., &LR., and &PB. to identify image copy data sets. For example, the following template name would make a meaningful seven-character data set qualifier:
  COPY&IC.&LR.&PB.
• Use &DS for FlashCopy image copies for uniqueness when copying table spaces or index spaces at the space level.

What to do next

You can check the data set names by using the PREVIEW function. In PREVIEW mode, Db2 expands all TEMPLATE data set names in the SYSIN DD, in addition to any data set name from the TEMPLATE DD that are referenced on a utility control statement. Db2 then prints the information to the SYSPRINT data set and halts execution. You can specify PREVIEW in one of two ways, either as a JCL PARM or on the OPTIONS PREVIEW utility control statement.

Related reference:
“Syntax and options of the TEMPLATE control statement” on page 793
“Syntax and options of the OPTIONS control statement” on page 407

Default space calculations for data set templates

Db2 calculates the space for data sets that are defined by the TEMPLATE utility based on the utility that is using the template. For disk data sets, all of this space is allocated as a primary quantity by default.

Data set size

For disk data sets, Db2 estimates the size of the data set based on formulas that vary according to the utility and the data set. These space estimation formulas are shown in the “Data sets that utility uses” topics for each online utility. Alternatively, you can specify your own values for disk space by using the SPACE option in the TEMPLATE utility control statement.

Db2 usually estimates the size of a data set based on the size of other existing data sets. However, if any of the required data sets are on tape, Db2 is unable to estimate the size.

When Db2 is able to calculate size, it calculates the maximum size. This action can result in overly large data sets. Db2 always allocates data set size with the RLSE (release) option so that unused space is released on deallocation. However in some cases, the calculated size of required data sets is too large for the DYNALLOC interface to handle. In this case, Db2 issues error message DSNU1034I, and you must allocate the data set by a DD statement. If the object is part of a LISTDEF list, you might need to remove it from the list and process it individually.
**Recommendation:** To improve the accuracy of the default space estimation, run the RUNSTATS utility with the UPDATE SPACE or UPDATE ALL option before you run any of the following utilities:

- CHECK DATA
- CHECK INDEX
- CHECK LOB
- REBUILD INDEX
- REORG INDEX
- REORG TABLESPACE
- UNLOAD

**Extent allocation for disk data sets**

By default, for data sets on disk, 100 percent of the required space that is estimated by Db2 is allocated as a primary quantity. If this amount of space is typically not available on a single volume, specify the PCTPRIME option with a value lower than 100. Alternatively, if you want the upper limit of the primary quantity based on size instead of percentage, use the MAXPRIME option.

After the primary space is allocated, a secondary quantity that is equal to the estimated required space is divided into the specified number of secondary extents. This number is identified by the NBRSECND option. Individual utilities might request larger secondary extents to compensate for localized uncertainty in the space estimations. If you specify either PCTPRIME or MAXPRIME, any secondary allocation requests are limited to the size of the primary allocation.

**Related reference:**

- “Syntax and options of the TEMPLATE control statement” on page 793
- Chapter 32, “RUNSTATS,” on page 741

**Related information:**

- DSNU1034I (Db2 Messages)

**Guidelines for templates and tape data sets**

When you use the TEMPLATE utility to allocate tape data sets, use the STACK option to control tape processing.

STACK NO specifies traditional, single-file processing. The data set is written, and the tape is rewound and repositioned or even remounted. STACK YES specifies that successive files are to be written on a single logical tape without repositioning or remounting.

When you specify STACK YES, Db2 has the following behavior:

- Db2 stacks files only within a single utility invocation. When that utility ends, the stack is terminated, which means that the tape is rewound and unloaded. To allow more stacking, use the LISTDEF utility to define a list and then specify that list in another utility control statement. Using a LISTDEF list forces multiple objects to be processed under a single utility invocation.

- To preserve parallel processing, parallel tasks are written to different tape volumes. The specific volume to which the data set is written can vary, depending on the following factors:
  - The number of output data sets that are being produced
  - The number of parallel processes that are requested
The number of tape units that are available to the job step

If you specify STACK YES, take the following actions as needed:

- To avoid issues with mounting a tape volume on the wrong drive, specify UNCNT 1.
- To prevent conflicts between parallel processes, use a single process to write a file to a stack. (Parallel processing can complicate stacking.)
- Ensure that only files of the same type are stacked on the same tape. For example, one tape might contain local primary image copies whereas another tape might contain remote primary image copies. The file types cannot be mixed.
- Ensure that the number of TEMPLATE utility control statements that specify STACK YES is no more than 85 for one invocation of the utility.

Restrictions: Do not use the STACK YES option in the following situations:

- For concurrent copies (copies that are made by the COPY utility with the CONCURRENT option)
- For inline image copies that are created by REORG TABLESPACE on a partition-by-growth base table space with one or more LOB columns
- For data sets that might be redirected to disk by automatic class selection (ACS) rules

The data sets and utilities for which the STACK YES option are supported are listed in the following table. “Yes” indicates that the specified utility supports tape stacking for the specified data set. “No” indicates that the specified utility does not support tape stacking for the specified data set. “Ignored” indicates that the specified data set does not apply to the specified utility.

Table 119. Supported data sets for tape stacking

<table>
<thead>
<tr>
<th>ddname</th>
<th>CHECK DATA</th>
<th>CHECK INDEX or CHECK LOB</th>
<th>COPY</th>
<th>COPY-TOCOPY</th>
<th>LOAD</th>
<th>MERGE-COPY</th>
<th>REBUILD INDEX</th>
<th>REORG INDEX</th>
<th>REORG TABLESPACE</th>
<th>UNLOAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSREC</td>
<td>ignored</td>
<td>ignored</td>
<td>ignored</td>
<td>ignored</td>
<td>No</td>
<td>ignored</td>
<td>ignored</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>SYSDISC</td>
<td>ignored</td>
<td>ignored</td>
<td>ignored</td>
<td>ignored</td>
<td>No</td>
<td>ignored</td>
<td>ignored</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>SYSPUNCH</td>
<td>ignored</td>
<td>ignored</td>
<td>ignored</td>
<td>ignored</td>
<td>ignored</td>
<td>ignored</td>
<td>ignored</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>SYSCOPY</td>
<td>ignored</td>
<td>ignored</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>ignored</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>SYSCOPY2</td>
<td>ignored</td>
<td>ignored</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>ignored</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>SYSRCPY1</td>
<td>ignored</td>
<td>ignored</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>ignored</td>
<td>Yes</td>
<td>ignored</td>
<td></td>
</tr>
<tr>
<td>SYSRCPY2</td>
<td>ignored</td>
<td>ignored</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>ignored</td>
<td>Yes</td>
<td>ignored</td>
<td></td>
</tr>
<tr>
<td>SYSUT1</td>
<td>No</td>
<td>No</td>
<td>ignored</td>
<td>ignored</td>
<td>No</td>
<td>ignored</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>ignored</td>
</tr>
<tr>
<td>SORTOUT</td>
<td>No</td>
<td>ignored</td>
<td>ignored</td>
<td>ignored</td>
<td>No</td>
<td>ignored</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>ignored</td>
</tr>
<tr>
<td>SYSMAP</td>
<td>ignored</td>
<td>ignored</td>
<td>ignored</td>
<td>ignored</td>
<td>No</td>
<td>ignored</td>
<td>ignored</td>
<td>ignored</td>
<td>ignored</td>
<td></td>
</tr>
<tr>
<td>SYSERR</td>
<td>No</td>
<td>ignored</td>
<td>ignored</td>
<td>ignored</td>
<td>No</td>
<td>ignored</td>
<td>ignored</td>
<td>ignored</td>
<td>ignored</td>
<td></td>
</tr>
<tr>
<td>FILTERDS</td>
<td>ignored</td>
<td>ignored</td>
<td>No</td>
<td>ignored</td>
<td>ignored</td>
<td>ignored</td>
<td>ignored</td>
<td>ignored</td>
<td>ignored</td>
<td></td>
</tr>
</tbody>
</table>

Related reference:

“Syntax and options of the TEMPLATE control statement” on page 793
Chapter 18, “LISTDEF,” on page 215
How TEMPLATE supports GDG data sets

When you use the TEMPLATE utility, you can specify both absolute version references and relative references to generation data groups (GDGs) in the DSN name operand.

The first time that the data set is referenced, Db2 detects the absence of a GDG base. When (+1) or some other parenthetical relative expression is used, Db2 creates the GDG base. By default, the new base has a limit of 99 entries. Use the GDGLIMIT keyword to alter this value or prohibit this action.

After the base is created, you can specify either the absolute version G0000V00 or a relative version. If you use the PREVIEW function on the OPTIONS utility control statement, Db2 displays the GDG relative version references. GDG names are restricted to 35 characters.

A model data set, as defined in the MODELDCB option, might be required to allocate GDG data sets in your environment.

Related reference:

“Syntax and options of the TEMPLATE control statement” on page 793
“Syntax and options of the OPTIONS control statement” on page 407

Template switching

Template switching is most commonly used to direct small data sets to disk and large data sets to tape, but it can also be used to switch to templates that differ in DSNs or in HSN classes. The decision to switch is made based on the estimated output data set size, which may differ from the actual final size of the output data set. This difference is particularly true for incremental image copies that are estimated at 10% of the space required for a full image copy.

Termination or restart of TEMPLATE

You can terminate and restart a TEMPLATE utility job.

You can terminate a TEMPLATE utility job by using the TERM UTILITY command if you submitted the job or have SYSOPR, SYSCTRL, or SYSADM authority.

You can restart a TEMPLATE utility job, but it starts from the beginning again.

If you are restarting this utility as part of a larger job in which TEMPLATE completed successfully, but a later utility failed, see the information about restarting utilities with templates.

Related concepts:

“How utilities restart with templates” on page 49

Related tasks:

“Restarting an online utility” on page 44

Sample TEMPLATE control statements

Use the sample control statements as models for developing your own TEMPLATE control statements.
Example 1: Specifying a basic template for an image copy on disk

The following TEMPLATE utility control statement defines a basic template that can be used to allocate an image copy data set. The name of the template is COPYDS. Any subsequent COPY jobs that specify this template for dynamically allocated data sets use the data set naming convention that is defined by the DSN option.

```
TEMPLATE COPYDS DSN &DB..&TS..COPY&IC..&LR..&PB..&DATE..&TIME.
```

Example 2: Using variable substring notation to specify data set names

The following control statement defines template CP2. Variable substring notation is used in the DSN option to define the data set naming convention.

Assume that in the year 2003 you make a full image copy of partition 00004 of table space DSN8S81D. Assume that you specify the template CP2 for the data set for the local primary copy. Db2 gives the following name to the image copy data set: DH173001.DSN8S81D.Y03.COPYLP.P004

Notice that every variable in the DSN option begins with an ampersand (&) and ends with a period (.). These ampersands and periods are not included in the data set name. Only periods that do not signal the end of a variable are included in the data set name.

```
TEMPLATE CP2 DSN 'DH173001.&SN..Y&YEAR(3)..COPY&LR..&PB..P&PART(3,3).'
UNIT(SYSDA)
```

Example 3: Using COPY with TEMPLATE with variable substring notation

The following TEMPLATE utility control statement defines template SYSCOPY. Variable substring notation is used in the DSN option to define the data set naming convention. The subsequent COPY utility control statement specifies that Db2 is to make a local primary copy of the first partition of table space DSN8D81A.DSN8S81E. COPY is to write this image copy to a data set that is dynamically allocated according to the SYSCOPY template. In this case, the resulting data set name is DSN8D81A.DSN8S81E.P001

```
TEMPLATE SYSCOPY DSN '&DB..&TS..P&PA(3).'
COPY TABLESPACE DSN8D81A.DSN8S81E DNUM 1 COPYDDN(SYSCOPY)
```

Notice that you can change the part variable in the DSN operand from P&PA(3) to P&PA(3,3). The resulting data set name is the same, because the length value of 3 is implied in the first specification.

Example 4: Specifying a template for tape data sets with an expiration date

The following control statement defines the TAPEDS template. Any data sets that are defined with this template are to be allocated on device number 3590-1, as indicated by the UNIT option, and are to expire on 1 January 2100, as indicated by the EXPDL option. The DSN option indicates that these data set names are to have the following three parts: database name, table space name, and date.

```
TEMPLATE TAPEDS DSN(&DB..&TS..&DATE.)
UNIT 3590-1 EXPDL '2100001'
```
Example 5: Specifying a disk template that gives space allocation parameters.

The following control statement defines the DISK template. Any data sets that are defined with this template are to have 100 cylinders of primary disk space and 10 cylinders of secondary disk space, as indicated by the SPACE and CYL options. The DSN option indicates that the data set names are to have the following three parts: database name, table space name, and time.

```
TEMPLATE DISK DSN &DB..&TS..T&TIME.
  SPACE(100,10) CYL
```

Example 6: Specifying a disk template that uses a default size with constraints

The following control statement defines the DISK template. Because the SPACE option does not specify quantities for primary and secondary space allocation, Db2 calculates these values with the following constraint: the maximum allowable primary space allocation is 1000 cylinders. This constraint is indicated by the MAXPRIME option. The DSN option indicates that the data set names are to have the following three parts: database name, table space name, and time.

```
TEMPLATE DISK DSN(&DB..&TS..T&TIME.)
  SPACE CYL MAXPRIME 1000
```

Example 7: Using TEMPLATE with LISTDEF and COPY

In the following example, the LISTDEF utility control statement defines the CPY1 list. The TEMPLATE control statement then defines the TMP1 template. The COPY utility control statement then specifies that Db2 is to make local copies of the objects in the CPY1 list. Db2 is to write these copies to data sets that are dynamically allocated according to the characteristics that are defined in the TMP1 template.

```
LISTDEF CPY1 INCLUDE TABLESPACES DBA906*.T*A906*
  INCLUDE INDEXSPACES COPY YES INDEXSPACE ADMF001.I?A906*
TEMPLATE TMP1 UNIT SYSDA
  DSN (DH109006..&STEPNAME..&SN..T&TIME.)
  DISP (MOD,CATLG,CATLG)
COPY LIST CPY1 COPYDDN (TMP1) PARALLEL (2) SHRLEVEL REFERENCE
```

Parentheses for the DSN name-expression are optional.

Example 8: Use TEMPLATE to create a GDG data set

In this example, the TEMPLATE control statement defines the COPYTEMP template. The COPY utility control statement specifies that Db2 is to write a local image copy of the table space DBLT2501.TPLT2501 to a data set that is dynamically allocated according to the characteristics that are defined in the COPYTEMP template. According to the COPYTEMP template, this data set is to be named JULTU225.GDG(+1) (as indicated by the DSN option) and is to have six entries created in the GDG base (as indicated by the GDGLIMIT option). The control block information is to be the same as that in the JULTU225.MODEL data set, as indicated by the MODELDCB option.
Example 9: Using a template to copy a GDG data set to tape

In this example, the OPTIONS utility control statement causes the subsequent TEMPLATE statement to run in PREVIEW mode. In this mode, Db2 checks the syntax of the TEMPLATE statement. If Db2 determines that the syntax is valid, it expands the data set names. The OPTIONS OFF statement ends PREVIEW mode processing. The subsequent COPY utility control statement executes normally. The COPY statement specifies that Db2 is to write a local image copy of the table space DBLT2501.TPLT2501 to a data set that is dynamically allocated according to the characteristics that are defined in the COPYTEMP template. According to the COPYTEMP template, this data set is to be named JULTU225.GDG(+1) (as indicated by the DSN option) and is to be stacked on the tape volume 99543 (as indicated by the UNIT, STACK, and VOLUMES options). The data set set dispositions are specified by the DISP option. The GDGLIMIT option specifies that 50 entries are to be created in a GDG base.

Figure 77. Example TEMPLATE and COPY statements for writing a local copy to a data set that is dynamically allocated according to the characteristics of the template.
Example 10: Creating a template that can be used for unloading LOB objects

The TEMPLATE control statement in this example defines a template called LOBFRV. The subsequent UNLOAD statement specifies that each CLOB in the RESUME column is to be unloaded to files that are dynamically allocated according to the characteristics defined for the LOBFRV template. In this case, those files are to be partitioned data sets, as specified by the DSNTYPE option. Each data set is to have the name UNLODTEST.database-name.LOB-table-space-name.RESUME, as specified by the DSN option. The names of each CLOB PDS is written to the unload data set. By default, the unload data set is defined by the SYSREC DD statement or template.

```
/*
//**************************************************
//** COMMENT: COPY GDG DATA SET TO TAPE
//**************************************************
//STEP1 EXEC DSNUPROC,UID='JULTU243.GDG',
//       UTPROC='',
//       SYSTEM='SSTR'
//SYSIN DD *
OPTIONS PREVIEW
   TEMPLATE COPYTEMP
   UNIT TAPE
   DSN 'JULTU243.GDG(+1)'
   VOLUMES (99543)
   GDGLIMIT(50)
   DISP(NEW,CATLG,CATLG)
   STACK YES
OPTIONS OFF
   COPY TABLESPACE DBLT4301.TPLT4301
       FULL YES
       COPYDDN (COPYTEMP)
       SHRLEVEL REFERENCE
/*
```

Figure 78. Example job that uses OPTIONS, TEMPLATE, and COPY statements to copy a GDG data set to tape.

Example 11: Using template switching.

The following TEMPLATE control statement assumes that tables space SMALL.TS occupies 10 cylinders and table space LARGE.TS occupies 100 cylinders. Both COPY statements use the SMALLTP template which specifies a limit of 20 cylinders. Table space SMALL.TS is smaller than this limit so no switching is performed. The output data set for table space SMALL.TS will be allocated on UNIT=SYSALLDA. Table space LARGE.TS is larger than this limit so the template is switched to the LARGETP template. The output data set for table space LARGE.TS will be allocated on UNIT=TAPE.

```
 TEMPLATE LOBFRV DSN 'UNLODTEST.&DB..&TS..RESUME'
             DSNTYPE(PDS) UNIT(SYSDA)

UNLOAD DATA
   FROM TABLE DSNB3910.EMP_PHOTO_RESUME
       (EMPNO CHAR(6),
       RESUME VARCHAR(255) CLOBF LOBFRV)
       SHRLEVEL CHANGE
```
TEMPLATE LARGETP DSN &DB..&TS..&DA..&TI.. UNIT=TAPE
TEMPLATE SMALLTP DSN &DB..&TS..&DA..&TI.. UNIT=SYSALLDA LIMIT( 20 CYL, LARGETP )
COPY TABLESPACE SMALL.TS COPYDDN( SMALLTP )
COPY TABLESPACE LARGE.TS COPYDDN( SMALLTP )
Chapter 35. UNLOAD

The UNLOAD online utility copies data from one or more source objects to one or more BSAM sequential data sets in external formats. The output records that the UNLOAD utility writes are compatible as input to the LOAD utility. Therefore, you can use this output to reload the original table or different tables.

Although the function of the UNLOAD utility is often referred to as unloading data, the data is not deleted from the source object. The utility just makes a copy of the data. That copy includes the data only; it does not include all of the pages, such as the system pages and header pages, that are included in an image copy.

The source for UNLOAD can be Db2 table spaces or Db2 image copy data sets. The source cannot be a concurrent copy or a FlashCopy image copy.

You can unload rows from an entire table space or select specific partitions or tables to unload. You can also select columns by using the field specification list. If a table space is partitioned, you can unload all of the selected partitions into a single data set. Alternatively, you can unload each partition in parallel into physically distinct data sets.

UNLOAD must be run on the system where the definitions of the table space and the table exist.

Output

UNLOAD generates an unloaded table space or partition.

Authorization required

To execute this utility, you must use a privilege set that includes one of the following authorities:
- Ownership of the tables
- SELECT privilege on the tables
- DBADM authority for the database. If the object on which the utility operates is in an implicitly created database, DBADM authority on DSNDB04 or the implicitly created database is sufficient.
- DATAACCESS authority
- SYSDM authority
- SYSCtrl authority (catalog tables only)
- SQLADM authority (catalog tables only)
- System DBADM authority (catalog tables only)
- ACCESSCTRL authority (catalog tables only)
- SECADM authority (catalog tables only)

If you use RACF access control with multilevel security and UNLOAD is to process a table space that contains a table that has multilevel security with row-level granularity, you must be identified to RACF and have an accessible valid security label. Each row is unloaded only if your security label dominates the data security label. If your security label does not dominate the data security label, the row is not unloaded, but Db2 does not issue an error message.
Restrictions on running UNLOAD

- UNLOAD cannot be run on a table space during the period after RECOVER is run to a point in time before materialization of pending definition changes and before REORG is run to complete the point-in-time recovery process.

Execution phases of UNLOAD

The UNLOAD utility operates in these phases:

1. UTILINIT initializes the environment.
2. UNLOAD unloads records to sequential data sets. One pass through the input data set is made. If UNLOAD is processing a table space or partition, Db2 takes internal commits. These commits provide commit points at which the utility can be restarted if the utility stops in this phase.
3. UTILTERM cleans up the environment.

Before running UNLOAD

If you plan to run UNLOAD on encrypted data, do not use the WHEN statement to filter encrypted fields; UNLOAD cannot filter rows that contain encrypted data.

If the table into which you are loading data has an index with these characteristics, the LOAD fails:
- The index was created on a VARBINARY column or a column with a distinct type that is based on a VARBINARY data type.
- The index column has the DESC attribute.

To fix the problem, drop the index, or alter the column data type to BINARY, and then rebuild the index.

Data sets that UNLOAD uses

The following table lists the data sets that UNLOAD uses. The table lists the DD name that is used to identify the data set, a description of the data set, and an indication of whether it is required. Include statements in your JCL for each required data set and any optional data sets that you want to use.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSIN</td>
<td>Input data set that contains the utility control statement.</td>
<td>Yes</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>Output data set for messages.</td>
<td>Yes</td>
</tr>
<tr>
<td>SYSPUNCH</td>
<td>One or more work data sets that contain the generated LOAD statements for subsequently reloading the data. The default DD name is PUNCHDDN.</td>
<td>No¹</td>
</tr>
<tr>
<td>Unload data set</td>
<td>One or more work data sets that contain the unloaded table rows. The default DD name is SYSREC.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Note:
1. Required if you request that UNLOAD generate LOAD statements by specifying PUNCHDDN in the utility control statement.
The following object is named in the utility control statement and does not require a DD statement in the JCL:

**Table space**

Table space that is to be unloaded. (If you want to unload only one partition of a table space, you must specify the PART option in the control statement.)

**Concurrency for UNLOAD**

Db2 treats individual data partitions as distinct source objects. Utilities that operate on different partitions of the same table space are compatible.

**Claims and drains for UNLOAD**

The following table shows which claim classes UNLOAD drains and the restrictive states that the utility sets.

*Table 121. Claim classes of UNLOAD operations*

<table>
<thead>
<tr>
<th>Target</th>
<th>UNLOAD</th>
<th>UNLOAD PART</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table space or physical partition of a table space with SHRLEVEL REFERENCE</td>
<td>DW/UTRO</td>
<td>DW/UTRO</td>
</tr>
<tr>
<td>Table space or physical partition of a table space with SHRLEVEL CHANGE</td>
<td>CR/UTRW</td>
<td>CR/UTRW</td>
</tr>
<tr>
<td>Image copy*</td>
<td>CR/UTRW</td>
<td>CR/UTRW</td>
</tr>
</tbody>
</table>

**Legend:**

- DW: Drain the write claim class, concurrent access for SQL readers
- UTRO: Utility restrictive state, read-only access allowed
- CR: Claim read, concurrent access for SQL writers and readers
- UTRW: Utility restrictive state; read-write access allowed

**Note:** * If the target object is an image copy, the UNLOAD utility applies CR/UTRW to the corresponding table space or physical partitions to prevent the table space from being dropped while data is being unloaded from the image copy, even though the UNLOAD utility does not access the data in the table space.

**Compatibility for UNLOAD**

The compatibility of the UNLOAD utility and the other utilities on the same target objects are shown in the following table. If the SHRLEVEL REFERENCE option is specified, only SQL read operations are allowed on the same target objects; otherwise SQL INSERT, DELETE, and UPDATE are also allowed. If the target object is an image copy, INSERT, DELETE, and UPDATE are always allowed on the corresponding table space. In any case, DROP or ALTER cannot be applied to the target object while the UNLOAD utility is running.

*Table 122. Compatibility of UNLOAD with other utilities*

<table>
<thead>
<tr>
<th>Action</th>
<th>UNLOAD SHRLEVEL REFERENCE</th>
<th>UNLOAD SHRLEVEL CHANGE</th>
<th>FROM IMAGE COPY</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHECK DATA DELETE NO</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CHECK DATA DELETE YES</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CHECK INDEX</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Table 122. Compatibility of UNLOAD with other utilities (continued)

<table>
<thead>
<tr>
<th>Action</th>
<th>UNLOAD SHRLEVEL REFERENCE</th>
<th>UNLOAD SHRLEVEL CHANGE</th>
<th>FROM IMAGE COPY</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHECK LOB</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COPY INDEXSPACE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COPY TABLESPACE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes*</td>
</tr>
<tr>
<td>DIAGNOSE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LOAD SHRLEVEL CHANGE</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LOAD SHRLEVEL NONE</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MERGECOPY</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>MODIFY RECOVERY</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>MODIFY STATISTICS</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>QUIESCE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>REBUILD INDEX</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RECOVER (no options)</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RECOVER ERROR RANGE</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RECOVER TOCOPY or TORBA</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REORG INDEX</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>REORG TABLESPACE UNLOAD CONTINUE or PAUSE</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REORG TABLESPACE UNLOAD ONLY or EXTERNAL</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>REPAIR DUMP or VERIFY</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>REPAIR LOCATE INDEX PAGE REPLACE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>REPAIR LOCATE KEY or RID DELETE or REPLACE</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REPAIR LOCATE TABLESPACE PAGE REPLACE</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REPORT</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RUNSTATS INDEX</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RUNSTATS TABLESPACE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>STOSPACE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Table 122. Compatibility of UNLOAD with other utilities (continued)

<table>
<thead>
<tr>
<th>Action</th>
<th>UNLOAD SHRLEVEL REFERENCE</th>
<th>UNLOAD SHRLEVEL CHANGE</th>
<th>FROM IMAGE COPY</th>
</tr>
</thead>
</table>

Note: If the same data set is used as the output from the COPY utility and as the input data set of the UNLOAD utility, unexpected results can occur.

UNLOAD utility syntax and options

See “Syntax and options of the UNLOAD control statement.”

Termination or restart of UNLOAD

If you terminate UNLOAD by using the TERM UTILITY command during the unload phase, the output records are not erased. The output data set remains incomplete until you either delete it or restart the utility job.

When the source is one or more table spaces, you can restart the UNLOAD job at the partition level or at the table space level when data is unloaded from multiple table spaces by using the LIST option. When you restart a terminated UNLOAD job, processing begins with the table spaces or partitions that had not yet been completed. For a table space or partitions that were being processed at termination, UNLOAD resets the output data sets and processes those table space or partitions again.

When the source is one or more image copy data sets (when FROMCOPY or FROMCOPYDDN is specified), UNLOAD always starts processing from the beginning.

Related concepts:

Multilevel security (Managing Security)

Syntax and options of the UNLOAD control statement

The UNLOAD utility control statement, with its multiple options, defines the function that the utility job performs.

You can create a control statement with the ISPF/PDF edit function. After creating it, save it in a sequential or partitioned data set. When you create the JCL for running the job, use the SYSIN DD statement to specify the name of the data set that contains the utility control statement.
Syntax diagram

source-spec:

Notes:
1 The FROMSEQNO option is required if you are unloading an image copy from a tape data set that is not cataloged.
FROM-TABLE-spec:

The syntax diagram and option descriptions for the FROM-TABLE-spec are presented in the section “FROM-TABLE-spec syntax diagram and option descriptions” on page 838.

Option descriptions

DATA

Identifies the data that is to be selected for unloading with table-name in the from-table-spec. The DATA keyword is mutually exclusive with TABLESPACE, PART, and LIST keywords.

When you specify the DATA keyword, or you omit either the TABLESPACE or the LIST keyword, you must also specify at least one FROM TABLE clause.
**TABLESPACE**

Specifies the table space (and, optionally, the database to which it belongs) from which the data is to be unloaded.

*database-name*

The name of the database to which the table space belongs. The name cannot be DSNDDB01 or DSNDDB07.

The default value is **DSNDDB04**.

*tablespace-name*

The name of the table space from which the data is to be unloaded. The specified table space must not be a LOB or XML table space.

**PART**

Identifies a partition or a range of partitions from which the data is to be unloaded. This keyword applies only if the specified table space is partitioned. You cannot specify PART with LIST. The maximum is 4096.

*integer*

Designates a single partition. *integer* must identify an existing physical partition number within the table space.

*int1:int2*

Designates a range of partitions from physical partition number *int1* to physical partition number *int2*. *int1* must be a positive integer that is less than the highest partition number within the table space. *int2* must be an integer that is greater than *int1* and less than or equal to the highest partition number.

If no PART keyword is specified in an UNLOAD control statement, the data from the entire table space is unloaded into a single unload data set.

**FROMCOPY** *data-set-name*

Indicates that data is to be unloaded from an image copy data set. When you specify FROMCOPY, the UNLOAD utility processes only the specified image copy data set. Alternatively, you can use the FROMCOPYDDN keyword where multiple image copy data sets can be concatenated under a single DD name.

*data-set-name*

Specifies the name of a single image copy data set.

Related information:

[“Unloading data from image copy data sets” on page 868](#)

**FROMVOLUME**

Identifies the volume where the image copy data set resides.

**CATALOG**

Indicates that the data set is cataloged. Use this option only for an image copy that was created as a cataloged data set (which means that its volume serial is not recorded in SYSIBM.SYSCOPY).

*vol-ser*

Identifies the data set by an alphanumeric volume serial identifier of its first volume. Use this option only for an image copy that was created as a non-cataloged data set. To specify a data set that is stored on multiple tape volumes, identify the first *vol-ser* in the SYSCOPY record.

**FROMSEQNO** *n*

Identifies the image copy data set by its file sequence number. The
FROMSEQNO option is required if you are unloading an image copy from a tape data set that is not cataloged.

n  Specifies the file sequence number.

FROMCOPYDDN  ddname
Indicates that data is to be unloaded from one or more image copy data sets that are associated with the specified ddname. Multiple image copy data sets (primarily for the copy of pieces) can be concatenated under a single DD name.

ddname
Identifies a DD name with which one or more image copy data sets are associated.

Related information:
“Unloading data from image copy data sets” on page 868

LIST  listdef-name
Identifies the name of a list of objects that are defined by a LISTDEF utility control statement. The list can include table spaces, index spaces, databases, a tables, an index, and partitions. The list cannot include index spaces, LOB table spaces, and directory objects. You cannot use the LIST option to specify image copy data sets.

When you specify the LIST option, the referenced LISTDEF identifies:
• The table spaces from which the data is to be unloaded. You can use the pattern-matching feature of LISTDEF.
• The partitions (if a table space is partitioned) from which the data is to be unloaded (defined by the INCLUDE, EXCLUDE, and PARTLEVEL keywords in the LISTDEF statement).

The UNLOAD utility associates a single table space with one output data set, except when partition-parallelism is activated. When you use the LIST option with a LISTDEF that represents multiple table spaces, you must also define a data set TEMPLATE that corresponds to all of the table spaces and specify the template-name in the UNLDDN option.

If you want to generate the LOAD statements, you must define another TEMPLATE for the PUNCHDDN data set that is similar to UNLDDN. Db2 then generates a LOAD statement for each table space. This utility will only process clone data if the CLONE keyword is specified. The use of CLONED YES on the LISTDEF statement is not sufficient.

The partitions or partition ranges can be specified in a list.

PUNCHDDN
Specifies the DD name for a data set or a template name that defines one or more data set names that are to receive the LOAD utility control statements that the UNLOAD utility generates.

ddname
  Specifies the DD name.
  The default value is SYSPUNCH.

template-name
  Identifies the name of a data set template that is defined by a TEMPLATE utility control statement.

If the specified name is defined both as a DD name (in the JCL) and as a template name (in a TEMPLATE statement), it is treated as the DD name.
When you run the UNLOAD utility for multiple table spaces and you want to generate corresponding LOAD statements, you must have multiple output data sets that correspond to the table spaces so that Db2 retains all of the generated LOAD statements. In this case, you must specify an appropriate template name to PUNCHDDN. If you omit the PUNCHDDN specification, the LOAD statements are not generated.

If the partition variable (&PART. or &PA.) is included in a TEMPLATE for PUNCHDDN, Db2 replaces the &PART. or &PA. variable with the lowest partition number in the list of partitions to be unloaded. The partition number is in the form nnnnn.

**UNLDDN**

Specifies the DD name for a data set or a template name that defines one or more data set names into which the data is to be unloaded.

<table>
<thead>
<tr>
<th>ddname</th>
<th>Specifies the DD name. The default value is SYSREC.</th>
</tr>
</thead>
</table>

| template-name | Identifies the name of a data set template that is defined by a TEMPLATE utility control statement. |

If the specified name is defined both as a DD name (in the JCL) and as a template name (in a TEMPLATE statement), it is treated as the DD name.

When you run the UNLOAD utility for a partitioned table space, the selected partitions are unloaded in parallel if the following conditions are true:

1. You specify a template name for UNLDDN.
2. The template data set name contains the partition as a variable (&PART. or &PA.) without substring notation. This template name is expanded into multiple data sets that correspond to the selected partitions.
3. The TEMPLATE control statement does not contain all of the following options:
   - STACK(YES)
   - UNIT(TAPE)
   - An UNCNT value that is less than or equal to one.

If conditions 1 and 2 are true, but condition 3 is false, partition parallelism is not activated and all output data sets are stacked on one tape.

Db2 cannot do parallel UNLOAD operations for partitions if you use substring notation for the partition variable (&PART. or &PA.) in the DSN argument, because the data set name might not be unique for all partitions. Therefore, Db2 sets the value of &PA to '00000', and uses a single UNLDDN data set for all partitions. This action might cause duplicate data set errors on subsequent UNLOAD jobs for other partitions of the same table space.

When you run the UNLOAD utility for multiple table spaces, the output records are placed in data sets that correspond to the respective table spaces. Therefore the output data sets must be physically distinctive, and you must specify an appropriate template name to UNLDDN. If you omit the UNLDDN specification, the SYSREC DD name is not used, and an error occurs.

If the partition variable (&PART. or &PA.) is included in the TEMPLATE DSN statement when partition parallelism is not applicable (when the source is a non-partitioned table space or an image copy), the variable is replaced by
'0000' in the actual data set name. In this case, warning message DSNU1252I is issued, and the UNLOAD utility issues return code 4.

**EBCDIC**

Specifies that all output data of the character type is to be in EBCDIC. If a different encoding scheme is used for the source data, the data (except for bit strings) is converted into EBCDIC.

If you do not specify EBCDIC, ASCII, UNICODE, or CCSID, the encoding scheme of the source data is preserved.

See the description of the CCSID option for this utility.

**ASCII**

Specifies that all output data of the character type is to be in ASCII. If a different encoding scheme is used for the source data, the data (except for bit strings) is converted into ASCII.

If you do not specify EBCDIC, ASCII, UNICODE, or CCSID, the encoding scheme of the source data is preserved.

See the description of the CCSID option for this utility.

**UNICODE**

Specifies that all output data of the character type (except for bit strings) is to be in Unicode. If a different encoding scheme is used for the source data, the data is converted into Unicode.

If you do not specify EBCDIC, ASCII, UNICODE, or CCSID, the encoding scheme of the source data is preserved.

See the description of the CCSID option of this utility.

**CCSID**

Specifies up to three coded character set identifiers (CCSIDs) that are to be used for the data of character type in the output records, including data that is unloaded in the external character formats.

`integer1` specifies the CCSID for SBCS data. `integer2` specifies the CCSID for mixed data. `integer3` specifies the CCSID for DBCS data. This option is not applied to data with a subtype of BIT.

If you specify both FORMAT DELIMITED and UNICODE, all output data is in CCSID 1208, UTF-8; any other specified CCSID is ignored.

The following specifications are also valid:

**CCSID**

- Indicates that only an SBCS CCSID is specified.
- Indicates that an SBCS CCSID and a mixed CCSID are specified.
- Specifies either a valid CCSID or 0.

If you specify a value of 0 for one of the arguments or omit a value, the encoding scheme that is specified by EBCDIC, ASCII, or UNICODE is assumed for the corresponding data type (SBCS, MIXED, or DBCS).

If you do not specify EBCDIC, ASCII, or UNICODE:
- If the source data is of character type, the original encoding scheme is preserved.
• For character strings that are converted from numeric, date, time, or timestamp data, the default encoding scheme of the table is used. For more information, see the CCSID option of the CREATE TABLE statement in

If you specify EBCDIC, ASCII, or UNICODE, the CCSIDs specified for SBCS, DBCS, and MIXED must be valid CCSIDs for the specified encoding scheme, or 0.

When a CCSID conversion is requested, CCSID character substitutions can occur in the output string. Use the NOSUBS option to prevent possible character substitutions during CCSID conversion.

NOSUBS

Specifies that CCSID code substitution is not to be performed during unload processing.

When a string is converted from one CCSID to another (including EBCDIC, ASCII, and Unicode), a substitution character is sometimes placed in the output string. For example, this substitution occurs when a character (referred to as a code point) that exists in the source CCSID does not exist in the target CCSID. You can use the NOSUBS keyword to prevent the UNLOAD utility from allowing this substitution.

If you specify the NOSUBS keyword and character substitution is attempted while data is being unloaded, this action is treated as a conversion error. The record with the error is not unloaded, and the process continues until the total error count reaches the number that is specified by MAXERR.

NOPAD

Specifies that the variable-length columns in the unloaded records are to occupy the actual data length without additional padding. As a result, the unloaded or discarded records might have varying lengths. If XML columns are unloaded without the use of file reference variables, NOPAD is the default.

When you do not specify NOPAD:
• Default UNLOAD processing pads variable-length columns in the unloaded records to their maximum length, and the unloaded records have the same length for each table.
• The padded data fields are preceded by the length fields that indicate the size of the actual data without the padding.
• When the output records are reloaded with the LOAD utility, padded data fields are treated as varying-length data.

If you specify DELIMITED, the NOPAD option is the default for variable-length columns. For fixed-length columns, the normal padding rules apply.

Although LOAD processes records with variable-length columns that are unloaded or discarded by using the NOPAD option, these records cannot be processed by applications that process only fields in fixed positions. For example, the LOAD statement that is generated for the EMP sample table would look similar to the LOAD statement in Figure 60 on page 586.

SPANNED

Indicates whether records are to be unloaded into a VBS data set in spanned record format.

YES

Records are to be unloaded in spanned record format.
When you specify SPANNED YES, the source object and UNLOAD statement must satisfy the following requirements:

- The source table must have at least one XML or LOB column. If the table does not have any XML or LOB columns, SPANNED YES is ignored.
- If you specify multiple FROM TABLE statements to unload multiple tables, all tables must contain at least one LOB or XML column. Otherwise, none of the tables are unloaded in spanned record format.
- The UNLOAD statement must include a field specification list that specifies all LOB and XML data at the end of the record. The data type must be specified for the LOB and XML columns. Do not include a length or POSITION value for the LOB and XML columns. If no field specifications are included or the data type is not included in the field specification, SPANNED YES is ignored.
- Do not specify the following options with SPANNED YES:
  - DELIMITED
  - FROMCOPY or FROMCOPYDDN
  - LIST (If you specify LIST, SPANNED YES is ignored.)

In the cases where SPANNED YES is ignored, the unload data set is created with VB (variable-length blocked) record format, which does not allow records that are greater than 32 KB.

When you specify SPANNED YES, UNLOAD also takes the following actions:

- Ignores the RECFM attribute of the data set.
- Uses the NOPAD option.
- Ignores the TRUNCATE option.
- Ignores large block interface (LBI) if used by any output data sets.

If you also specify PUNCHDDN, the generated LOAD statement lists the LOB and XML data in a field specification list in the same order as the UNLOAD field specification list.

**NO** Records are not to be unloaded in spanned record format.

### Related information:

- "Unloading data in spanned record format" on page 866
- Large Block Interface (LBI) (z/OS DFSMS Using Data Sets)

### FORMAT INTERNAL

Specifies that the output record format is Db2 internal format. UNLOAD does no field procedure processing, data conversion, or CCSID conversion on the data. If the UNLOAD control statement contains a field specification, it is ignored.

When FORMAT INTERNAL is specified:

- UNLOAD does not unload data for LOB or XML columns. UNLOAD issues a warning message that indicates that LOB or XML data was not unloaded.
- UNLOAD does not add trailing blanks to output from variable-length columns.
- UNLOAD decompresses the data and does decoding that is specified by edit procedures.
- UNLOAD ignores any field specifications in the UNLOAD utility control statement.
Restrictions:

- Data that is unloaded with FORMAT INTERNAL should be loaded only into the same table, or into a table that exactly matches the unloaded table definition, including having the same field procedures.

- FORMAT INTERNAL cannot be specified with any of the following options:
  - ASCII
  - CCSID
  - DECIMAL
  - DELIMITED
  - EBCDIC
  - FLOAT
  - HEADER
  - NOPAD
  - NOSUBS
  - UNICODE

DELIMITED

Specifies that the output data file is in a delimited format. When data is in a delimited format, all fields in the output data set are character strings or external numeric values. In addition, each column in a delimited file is separated from the next column by a column delimiter character.

For each of the delimiter types that you can specify, you must ensure that the delimiter character is specified in the code page of the target data. The delimiter character can be specified as either a character or hex constant. For example, to specify # as the delimiter, you can specify either COLDEL '#' or COLDEL X'23'. If the utility statement is coded in a character type that is different from the output file, such as a utility statement that is coded in EBCDIC and output data that is in Unicode, specify the delimiter character in the utility statement as a hex constant, or the result is unpredictable.

You cannot specify the same character for more than one type of delimiter (COLDEL, CHARDEL, and DECPT).

If you specify the FORMAT DELIMITED option:

- You cannot specify HEADER CONST or use any of the multiple FROM TABLE statements.
- UNLOAD ignores any specified POSITION statements within the UNLOAD utility control statement.

For delimited output, UNLOAD does not add trailing padded blanks to variable-length columns, even if you do not specify the NOPAD option. For fixed-length columns, the normal padding rules apply. For example, if a VARCHAR(10) field contains ABC, UNLOAD DELIMITED unloads the field as "ABC". However, for a CHAR(10) field that contains ABC, UNLOAD DELIMITED unloads it as "ABC". For information about using delimited output and delimiter restrictions, see "Unloading delimited files" on page 876. For more information about delimited files see Appendix G, "Delimited file format," on page 1125.

COLDEL

Specifies the column delimiter that is used in the output file. The default is a comma (,). For most ASCII and UTF-8 data, this value is X'2C', and for most EBCDIC data, this value is X'6B'.

CHARDEL

Specifies the character string delimiter that is used in the output file. The
default is a double quotation mark ("). For most ASCII and UTF-8 data, this value is X'22', and for most EBCDIC data, this value is X'7F'.

The UNLOAD utility adds the CHARDEL character before and after every character string. To delimit character strings that contain the character string delimiter, the UNLOAD utility repeats the character string delimiter where it used in the character string. The LOAD utility then interprets any pair of character delimiters that are found between the enclosing character delimiters as a single character. For example, the phrase what a "nice warm" day is unloaded as "what a "nice warm" day", and LOAD interprets it as what a "nice warm" day. The UNLOAD utility recognizes these character pairs for only CHAR, VARCHAR, and CLOB fields.

**DECPT**

Specifies the decimal point character that is used in the output file. The default is a period (.). For most ASCII and UTF-8 data, this value is X'2E', and for most EBCDIC data, this value is X'4B'.

**FLOAT**

Specifies the output format of the numeric floating-point data. This option applies to the binary output format only.

**S390**

Indicates that the binary floating point data is written to the output records in the S/390® internal format (also known as the hexadecimal floating point, or HFP).

**IEEE**

Indicates that the binary floating-point data is written to the output records in the IEEE format (also known as the binary floating point, or BFP).

**MAXERR** integer

Specifies the maximum number of records in error that are to be allowed; the unloading process terminates when this value is reached.

integer

Specifies the number of records in error that are allowed. When the error count reaches this number, the UNLOAD utility issues message DSNU1219 and terminates with return code 8.

The default value is 1, which indicates that UNLOAD stops when the first error is encountered. If you specify 0 or any negative number, execution continues regardless of the number of records that are in error.

If multiple table spaces are being processed, the number of records in error is counted for each table space. If the LIST option is used, you can add OPTION utility control statement (EVENT option with ITEMERROR) before the UNLOAD statement to specify that the table space in error is to be skipped and the subsequent table spaces are to be processed.

The MAXERR option is ignored when the UNLOAD utility encounters errors that prevent it from continuing to process data. For example, if you receive message DSNU283I, SQLCODE -452, and reason code 7 when unloading LOB or XML data using file reference variables, the UNLOAD utility terminates regardless of what you specified for MAXERR.

**SHRLEVEL**

Specifies whether other processes can access or update the table space or partitions while the data is being unloaded.
UNLOAD ignores the SHRLEVEL specification when the source object is an image copy data set.

The default value is **SHRLEVEL CHANGE ISOLATION CS**.

**CHANGE**

Specifies that rows can be read, inserted, updated, and deleted from the table space or partition while the data is being unloaded.

**ISOLATION**

Specifies the isolation level with SHRLEVEL CHANGE.

- **CS** Indicates that the UNLOAD utility is to read rows in cursor stability mode. With CS, the UNLOAD utility assumes CURRENTDATA(NO).

- **UR** Indicates that uncommitted rows, if they exist, are to be unloaded. The unload operation is performed with minimal interference from the other Db2 operations that are applied to the objects from which the data is being unloaded.

**SKIP LOCKED DATA**

Specifies that the UNLOAD utility is to skip rows on which incompatible locks are held by other transactions. This option applies to a row level or page level lock.

**REFERENCE**

Specifies that during the unload operation, rows of the tables can be read, but cannot be inserted, updated, nor deleted by other Db2 threads.

When you specify SHRLEVEL REFERENCE, the UNLOAD utility drains writers on the table space from which the data is to be unloaded. When data is unloaded from multiple partitions, the drain lock is obtained for all of the selected partitions in the UTILINIT phase.

**DECFLOAT_ROUNDMODE**

Specifies the rounding mode to be used when DECFLOATs are manipulated. The following rounding modes are supported:

- **ROUND_CEILING**
  Round toward +infinity. The discarded digits are removed if they are all zero or if the sign is negative. Otherwise, the result coefficient should be incremented by 1 (rounded up).

- **ROUND_DOWN**
  Round toward 0 (truncation). The discarded digits are ignored.

- **ROUND_FLOOR**
  Round toward -infinity. The discarded digits are removed if they are all zero or positive. Otherwise, the sign is negative and the result coefficient should be incremented by 1 (rounded up).

- **ROUND_HALF_DOWN**
  Round to the nearest number. If equidistant, round down. If the discarded digits are greater than 0.5, the result coefficient should be incremented by 1 (rounded up). The discarded digits are ignored if they are 0.5 or less.

- **ROUND_HALF_EVEN**
  Round to the nearest number. If equidistant, round so that the final digit is even. If the discarded digits are greater than .05, the result coefficient should be incremented by 1 (rounded up). The discarded digits are ignored if they are less than 0.5. If the result coefficient is .05 and the rightmost
digit is even, the result coefficient is not altered. If the result coefficient is .05 and the rightmost digit is odd, the result coefficient should be incremented by 1 (rounded up).

**ROUND_HALF_UP**
Round to nearest. If equidistant, round up. If the discarded digits are greater than or equal to 0.5, the result coefficient should be incremented by 1 (rounded up). Otherwise the discarded digits are ignored.

**ROUND_UP**
Round away from 0. If all of the discarded digits are 0, the result is unchanged. Otherwise, the result coefficient should be incremented by 1 (rounded up).

If the user does not specify DECFLOAT_ROUNDING_MODE, the default value of the DECFLOAT_ROUNDING_MODE option is DECFLOAT ROUNDING MODE from the DECP.

**IMPLICIT_TZ**
Specifies the implicit time zone to use when timestamp values are being unloaded from a TIMESTAMP column with no time zone, and the field specification for the column is TIMESTAMP WITH TIME ZONE EXTERNAL.

'timezone-string'
Specifies the implicit time zone value. The time zone is the difference (in hours and minutes) between local time and UTC. The range of the hour component is -12 to 14, and the minute component is 00 to 59. The time zone is specified in the form ±th:tm, with values ranging from -12:59 to +14:00.

IMPLICIT_TZ is a required keyword when the unload timestamp without time zone column to a timestamp with time zone column is used.

**PARALLEL**
Specifies the maximum number of subtasks that are to be used in parallel to process the unloading of a partitioned table space. If the PARALLEL keyword is omitted, the maximum number of subtasks is limited by the number of partitions being unloaded.

(num-subtasks)
Specifies the maximum number of subtasks that are to be processed in parallel. The value must be an integer between 0 and 32767, inclusive. If the specified value for num-subtasks is greater than 32767, the UNLOAD statement fails. If 0 or no value is specified for num-subtasks, the UNLOAD utility uses the optimal number of parallel subtasks after applying constraints. If the specified value for num-subtasks is greater than the calculated optimal number, the UNLOAD utility limits the number of parallel subtasks to the optimal number.

The specified number of subtasks for PARALLEL always overrides the specification of the PARAMDEG_UTIL subsystem parameter, so PARALLEL can be smaller or larger than the value of PARAMDEG_UTIL.

**CLONE**
Indicates that UNLOAD is to unload data from only clone tables in the specified table spaces. This utility will only process clone data if the CLONE keyword is specified. The use of CLONE YES on the LISTDEF statement is not sufficient. If you specify the name of the clone table in the FROM TABLE clause, you do not need to specify the CLONE keyword.
FROM-TABLE-spec syntax diagram and option descriptions

More than one table or partition for each table space can be unloaded with a single invocation of the UNLOAD utility. One FROM TABLE statement for each table that is to be unloaded is required to identify:

- A table name from which the rows are to be unloaded
- A field to identify the table that is associated with the rows that are to be unloaded from the table by using the HEADER option
- Sampling options for the table rows
- A list of field specifications for the table that is to be used to select columns that are to be unloaded
- Selection conditions, specified in the WHEN clause, that are to be used to qualify rows that are to be unloaded from the table

All tables that are specified by FROM TABLE statements must belong to the same table space. If rows from specific tables are to be unloaded, a FROM TABLE clause must be specified for each source table. If you do not specify a FROM TABLE clause for a table space, all the rows of the table space are unloaded.

Use a list of field specifications to specify the following characteristics:

- Column selection. Specifies the column names of a table that is to be unloaded. If a list of field specifications is given, only the listed columns are unloaded.
- Column ordering. Specifies the order of fields that are to be placed in the output records. If a list of field specifications is given, data of the listed columns is unloaded in the order of listed column names.
- Output field attributes and format. Specifies the data type, length, and format of the data in the output records.

If you omit a list of field specifications, all columns of the source table are unloaded in the defined column order for the table. The default output field types that correspond to the data types of the columns are used.

When unloading XML or LOB columns to a VBS data set, the LOB and XML values are written at the end of the record in their column definition order, as specified by the required field specification list. This order is the same order that the LOAD utility uses when reading XML and LOB values from a VBS data set.

In a FROM TABLE clause, you can use parentheses in only two situations: to enclose the entire field selection list, and in a WHEN selection clause. This usage avoids potential conflict between the keywords and field-names that are used in the field selection list. A valid sample of a FROM TABLE clause specification follows:

```
UNLOAD ...
  FROM TABLE tablename SAMPLE x (c1,c2) WHEN (c3>0)
```

You cannot specify FROM TABLE if the LIST option is already specified.
FROM-TABLE-spec

field-specification:
strip spec:

Notes:
1. If you specify VARGRAPHIC, BINARY, or VARBINARY, you cannot specify 'strip-char'. You can specify only X'strip-char'.

timestamp with time zone spec:

decfloat spec:

selection condition:

predicate:
basic predicate:

BETWEEN predicate:

IN predicate:

LIKE predicate:

NULL predicate:
labeled-duration-expression:

```
current_date
current_timestamp
with time zone
+ constant
- year
  - years
  - months
  - days
  - hours
  - minutes
  - seconds
  - microseconds
```

**Option descriptions for FROM TABLE**

*table-name*

Identifies a Db2 table from which the rows are to be unloaded and to which the options in the FROM TABLE clause are to be applied.

If the table name is not qualified by a schema name, the authorization ID of the invoker of the utility job step is used as the schema qualifier of the table name. Enclose the table name in quotation marks if the name contains a blank.

If you specify a dropped table on the FROM TABLE option, the UNLOAD utility terminates with return code 4.

**HEADER**

Specifies a constant header field, at the beginning of the output records, that can be used to associate an output record with the table from which it was unloaded.

If you specify a header field, it is used as the field selection criterion of the WHEN clause (a part of the INTO-TABLE specification) in the LOAD statement that is generated.

**OBID**

Specifies that the object identifier (OBID) for the table (a 2-byte binary value) is to be placed in the first 2 bytes of the output records that are unloaded from the table.

If you omit the HEADER option, **HEADER OBID** is the default, except for delimited files.

With HEADER OBID, the first 2 bytes of the output record cannot be used by the unloaded data. For example, consider the following UNLOAD statement:
UNLOAD ...
   FROM TABLE table-name HEADER OBID ...

The preceding UNLOAD statement generates a LOAD statement that is similar to the following example:

LOAD ...
   INTO TABLE table-name WHEN (1:2)=X'hh' ...

In this example, X'hh' is the hexadecimal notation of the OBID of table table-name.

NONE

Indicates that no record header field is to be created. HEADER NONE is the default value for a delimited file.

If HEADER NONE is specified in a FROM TABLE clause, the corresponding INTO TABLE clause in the generated LOAD statement does not have a WHEN specification. Therefore, if rows from multiple tables are unloaded and HEADER NONE is specified in one or more FROM TABLE clauses, rows that are unloaded from those tables are not able to be reloaded until you edit the generated LOAD statement. If you use the generated statement directly with the LOAD utility, the results might be unpredictable.

CONST

Specifies that a constant string is to be used as the record header. The given string operand determines the length of the header field. The string value must be enclosed by a pair of single quote characters.

For example, consider the following UNLOAD statement:

UNLOAD ...
   FROM TABLE table-name HEADER CONST 'abc' ...

The preceding UNLOAD statement generates a LOAD statement that is similar to the following example:

LOAD ...
   INTO TABLE table-name WHEN (1:3)='abc' ...

In this example, the given string is assumed to be in SBCS EBCDIC format. The output string of the HEADER field is in the specified or the default encoding scheme. If the encoding scheme that is used for output is not EBCDIC, the SBCS CCSID conversion is applied to the given string before it is placed in the output records. If the output SBCS encoding scheme is not EBCDIC, the WHEN condition in the generated LOAD statement contains a hexadecimal string.

You can also use the hexadecimal form, X'hex-string', to represent a string constant. If you want to specify a CONST string value in an encoding scheme other than SBCS EBCDIC, use the hexadecimal form. No CCSID conversion is performed if the hexadecimal form is used.

SAMPLE decimal

Indicates that only sampled rows of the table are to be unloaded. If selection conditions are specified by a WHEN clause within the same FROM TABLE clause, sampling is applied to the rows that are qualified by the WHEN selection conditions.
**decimal**
Specifies the percentage of the rows that are to be sampled in the decimal format. The precision is \textit{ddd.dddd}, and the valid range is \(0 \leq \text{decimal} \leq 100\).

If the number of rows to which the sampling is to be applied is \(N\):
- \(\text{decimal} \times N / 100\) rows are unloaded. (The fraction might be rounded to the nearest whole number.)
- If \(\text{decimal} > 0\) and \(N > 0\), at least one row is unloaded.
- If \(\text{decimal} = 100\), all rows from the table are unloaded.
- If the given \(\text{decimal} = 0\) or \(N = 0\), no row is unloaded from the table.

The sampling is applied for each individual table. If the rows from multiple tables are unloaded with sampling enabled, the referential integrity between the tables might be lost.

**LIMIT** \textit{integer}
Specifies the maximum number of rows that are to be unloaded from a table. If the number of unloaded rows reaches the specified limit, message DSNU1201 is issued for the table, and no more rows are unloaded from the table. The process continues to unload qualified rows from the other tables.

When partition parallelism is activated, the LIMIT option is applied to each partition instead of to the entire table.

\textit{integer}
Indicates the maximum number of rows that are to be unloaded from a table. If the specified number is less than or equal to zero, no row is unloaded from the table.

Like the SAMPLE option, if multiple tables are unloaded with the LIMIT option, the referential integrity between the tables might be lost.

**field-name**
Identifies a column name that must exist in the source table.

**POSITION**(start)
Specifies the field position in the output record. You can specify the position parameter as follows:

* An asterisk, indicating that the field starts at the first byte after the last position of the previous field.

\textit{start}  A positive integer that indicates the start column of the data field.

The default value is \textbf{POSITION(*)}.

The first column (byte position) of an output record corresponds to POSITION(1). If you specify HEADER NONE in the FROM TABLE clause, the item that is specified by the HEADER option is placed at the beginning of all the records that are unloaded from the table. You must account for the space for the record header:

- HEADER OBID (the default case): 2 bytes from position 1.
- HEADER CONST 'string' or \textit{X‘hex-string’} case: The length of the given string from position 1.

If the source table column can be null, the utility places a NULL indicator byte at the beginning of the data field in the output record. For BLOBF, CLOBF, or DBCLOBF columns, null values are indicated by a byte at the beginning of the file name. The \textit{start} parameter (or *) points to the position of the NULL
indicator byte. In the generated LOAD statement, \textit{start} is shifted by 1 byte to the right (as \textit{start}+1) so that, in the LOAD statement, the start parameter of the \texttt{POSITION} option points to the next byte past the NULL indicator byte.

For a varying-length field, a length field precedes the actual data field (after the NULL indicator byte, if applicable). For BLOBF, CLOBF, or DBCLOBF columns, the length of the file name is indicated by two bytes at the beginning of the file name. If the value cannot be null, the \textit{start} parameter (or *) points to the first byte of the length field. The size of the length field is either 4 bytes (BLOB, CLOB, or DBCLOB) or 2 bytes (VARCHAR or VARGRAPHIC).

When you explicitly specify the output field positions by using \textit{start} parameters (or using the * format) of the \texttt{POSITION} option, you must consider the following items as a part of the output field:

- For a field whose value can be null, a space for the NULL indicator byte
- For varying-length data, a space for the length field (either 2 bytes or 4 bytes)

“Layout of output fields” on page 873 illustrates the field layout in conjunction with the \texttt{POSITION} option, NULL indicator byte, the length field for a varying-length field, the \textit{length} parameter, and the actual data length.

The \texttt{POSITION} option is useful when the output fields must be placed at specific positions in the output records. The use of the \texttt{POSITION} parameters, however, can restrict the size of the output data fields. Use care when explicitly specifying \textit{start} parameters for nullable and varying-length fields. The TRUNCATE option might be required, if applicable, to fit a data item in a shorter space in an output record.

If you omit the \texttt{POSITION} option for the first field, the field starts from position 1 if HEADER NONE is specified. Otherwise, the field starts from the next byte position past the record header field. If \texttt{POSITION} is omitted for a subsequent field, the field is placed next to the last position of the previous field without any gap.

If NOPAD is specified and \texttt{POSITION} parameters are given for certain fields, the effect of the NOPAD option might be lost because the fields with \textit{start} parameters (other than the default *) always start at the fixed positions in the output records.

The \texttt{POSITION} option is ignored for delimited output files.

\textbf{CHAR}

Indicates that the output field is a character type with fixed length. You can use \texttt{CHARACTER} in place of \texttt{CHAR}. If the source table column can be null, a NULL indicator byte is placed at the beginning of the output field for a non-delimited output file.

If you specify the EBCDIC, ASCII, UNICODE, or CCSID options, the output data that corresponds to the specified option, is encoded in the CCSID, depending on the subtype of the source data (SBCS or MIXED). If the subtype is BIT, no conversion is applied.

\texttt{(length)}

Specifies the size of the output data in bytes.

- If followed by BLOBF, CLOBF, or DBCLOBF, the length specifies the size of the expanded template name in bytes.
- If the \textit{length} parameter is omitted, the default is the maximum length that is defined on the source table column or the length in bytes of the
expanded template name if BLOBF, CLOBF, or DBCLOBF follows the CHAR keyword. When the length parameter is specified:

- If the length is less than the size of the table column, the data is truncated to the length if the TRUNCATE keyword is present; otherwise, a conversion error occurs.
- For the case where BLOBF, CLOBF, or DBCLOBF immediately follows, an error will occur if the length is less than the size of the expanded template name.
- If the length is larger than the size of the table column, the output field is padded by the default pad characters to the specified length.

**BLOBF**

Specifies that the output field is to contain the name of the file to which the BLOB or XML is to be unloaded without CCSID conversion.

**BINAR YXML**

Specifies that the XML document is to be unloaded using file reference variables in Extensible Dynamic Binary XML Db2 Client/Server Binary XML Format (binary XML) format. This option is only supported when unloading an XML column.

**CLOBF**

Specifies that the output field is to contain the name of the file to which the CLOB or XML is to be unloaded with any required CCSID conversion.

**DBCLOBF**

Specifies that the output field is to contain the name of the file to which the DBCLOBF or XML is to be unloaded with any required CCSID conversion.

**TRUNCATE**

Indicates that a character string (encoded for output) is to be truncated from the right, if the data does not fit in the available space for the field in the output record. Truncation occurs at the character boundary. See “Specifying TRUNCATE and STRIP options for output data” on page 879 for the truncation rules that are used in the UNLOAD utility. Without TRUNCATE, an error occurs when the output field size is too small for the data.

**VARCHAR**

Specifies that the output field type is character of varying length. A 2-byte binary field indicating the length of data in bytes is prepended to the data field. If the table column can be null, a NULL indicator byte is placed before this length field for a non-delimited output file.

If you specify the EBCDIC, ASCII, UNICODE, or CCSID options, the output data is encoded in the CCSID corresponding to the specified option, depending on the subtype of the source data (SBCS or MIXED). If the subtype is BIT, no conversion is applied.

**(length)**

Specifies the maximum length of the actual data field in bytes. If you also specify NOPAD, it indicates the maximum allowable space for the data in the output records; otherwise, the space of the specified length is reserved for the data.

If followed by BLOBF, CLOBF, or DBCLOBF, length specifies the size of the expanded template name in bytes. If the length is less than the size of the expanded template name an error will occur.
If the length parameter is omitted, the default is the smaller of 255 and the maximum length that is defined on the source table column.

**BLOBF**
Specifies that the output field is to contain the name of the file to which the BLOB or XML is to be unloaded without CCSID conversion.

**BINARXML**
Specifies that the XML document is to be unloaded using file reference variables in binary XML format. This option is only supported when unloading an XML column.

**CLOBF**
Specifies that the output field is to contain the name of the file to which the CLOB or XML is to be unloaded with any required CCSID conversion.

**DBCLOBF**
Specifies that the output field is to contain the name of the file to which the DBCLOBF or XML is to be unloaded with any required CCSID conversion.

**STRIP**
Specifies that UNLOAD is to remove binary zeroes (the default) or the specified string from the beginning, the end, or both ends of the data. UNLOAD adjusts the VARCHAR length field (for the output field) to the length of the stripped data.

The STRIP option is applicable if the subtype of the source data is BIT. In this case, no CCSID conversion is performed on the specified strip character (even if it is given in the form ‘strip-char’).

The effect of the STRIP option is the same as the SQL STRIP scalar function.

**BOTH**
Indicates that UNLOAD is to remove occurrences of blank or the specified strip character from the beginning and end of the data. The default is **BOTH**.

**TRAILING**
Indicates that UNLOAD is to remove occurrences of blank or the specified strip character from the end of the data.

**LEADING**
Indicates that UNLOAD is to remove occurrences of blank or the specified strip character from the beginning of the data.

‘strip-char’
Specifies a single-byte character that is to be stripped. Specify this character value in EBCDIC. Depending on the output encoding scheme, UNLOAD applies SBCS CCSID conversion to the strip-char value before it is used in the strip operation. If you want to specify a strip-char value in an encoding scheme other than EBCDIC, use the hexadecimal form. UNLOAD does not perform CCSID conversion if the hexadecimal form is used.

*X'strip-char’*
Specifies a single-byte character that is to be stripped. It can be specified in the hexadecimal form, X‘hex-string’, where hex-string is two hexadecimal characters that represent a single SBCS character. If the strip-char operand is omitted, the default is the blank character, which is coded as follows:

- X‘40’, for the EBCDIC-encoded output case
• X'20' for the ASCII-encoded output case
• X'20' the Unicode-encoded output case

The strip operation is applied after the character code conversion, if the output character encoding scheme is different from the one that is defined on the source data. Therefore, if a strip character is specified in the hexadecimal format, you must specify the character in the encoding scheme that is used for output.

**TRUNCATE**
Indicates that a character string (encoded for output) is to be truncated from the right, if the data does not fit in the available space for the field in the output records. Truncation occurs at a character boundary. See “Specifying TRUNCATE and STRIP options for output data” on page 879 for the truncation rules that are used in the UNLOAD utility. Without TRUNCATE, an error occurs when the output field size is too small for the data.

**GRAPHIC**
Specifies that the output field is of the fixed-length graphic type. If the table column can be null, a NULL indicator byte is placed before the actual data field for any non-delimited output file.

If the output is in EBCDIC, the shift-in and shift-out characters are not included at the beginning and at the end of the data.

(length)
Specifies the number of DBCS characters (the size of the output data in bytes is twice the given length). If the given length is larger than the source data length, the output field is padded with the default pad character.

**TRUNCATE**
Indicates that a graphic character string (encoded for output) is to be truncated from the right, if the data does not fit in the available space for the field in the output records. Truncation occurs at a character (DBCS) boundary. Without TRUNCATE, an error occurs when the output field size is too small for the data.

**GRAPHIC EXTERNAL**
Specifies that the data is to be written in the output records as a fixed-length field of the graphic type with the external format; that is, the shift-out (SO) character is placed at the starting position, and the shift-in (SI) character is placed at the ending position. The byte count of the output field is always an even number.

GRAPHIC EXTERNAL is supported only in the EBCDIC output mode (by default or when the EBCDIC keyword is specified).

If the start parameter of the POSITION option is used to specify the output column position, it points to the (inserted) shift-out character at the beginning of the field. The shift-in character is placed at the next byte position past the last double-byte character of the data.

(length)
Specifies a number of DBCS characters, excluding the shift characters (as in the graphic type column definition that is used in a CREATE TABLE statement) nor the NULL indicator byte if the source column can be null. If the length parameter is omitted, the default output field size is the length that is defined on the corresponding table column, plus two bytes (shift-out and shift-in characters).
If the specified length is larger than the size of the data, the field is padded on the right with the default DBCS padding character.

**TRUNCATE**
Indicates that a graphic character string is to be truncated from the right by the DBCS characters, if the data does not fit in the available space for the field in the output records. Without TRUNCATE, an error occurs when the output field size is too small for the data. An error can also occur with the TRUNCATE option if the available space is less than 4 bytes (4 bytes is the minimum size for a GRAPHIC EXTERNAL field; shift-out character, one DBCS, and shift-in character); or fewer than 5 bytes if the field is can be null (the 4 bytes plus the NULL indicator byte).

**VARGRAPHIC**
Specifies that the output field is to be of the varying-length graphic type. A 2-byte binary length field is prepended to the actual data field. If the table column can be null, a NULL indicator byte is placed before this length field for any non-delimited output file.

(length)
Specifies the maximum length of the actual data field in the number of DBCS characters. If you also specify NOPAD, it indicates the maximum allowable space for the data in the output records; otherwise, the space of the specified length is reserved for the data.

If the length parameter is omitted, the default is the smaller of 127 and the maximum defined length of the source table column.

**STRIP**
Indicates that UNLOAD is to remove binary zeroes (the default) or the specified string from the unloaded data. UNLOAD adjusts the VARGRAPHIC length field (for the output field) to the length of the stripped data (the number of DBCS characters).

The effect of the STRIP option is the same as the SQL STRIP scalar function.

**BOTH**
Indicates that UNLOAD is to remove occurrences of blank or the specified strip character from the beginning and end of the data. The default is BOTH.

**TRAILING**
Indicates that UNLOAD is to remove occurrences of blank or the specified strip character from the end of the data.

**LEADING**
Indicates that UNLOAD is to remove occurrences of blank or the specified strip character from the beginning of the data.

X'strip-char'
Specifies a DBCS character that is to be stripped in the hexadecimal format, X'hhhh', where hhhh is four hexadecimal characters that represent a DBCS character. If this operand is omitted, the default is a DBCS blank in the output encoding scheme (for example, X'4040' for the EBCDIC-encoded output or X'8140' for CCSID 301).

The strip operation is applied after the character code conversion, if the output character encoding scheme is different from the one that is defined on the source data. Therefore, if you specify a strip character, it must be in the encoding scheme that is used for the output.
TRUNCATE
Indicates that a graphic character string (encoded for output) is to be truncated from the right, if the data does not fit in the available space for the field in the output records. Truncation occurs at a DBCS character boundary. Without TRUNCATE, an error occurs when the output field size is too small for the data.

SMALLINT
Specifies that the output field is a 2-byte binary integer (a negative number is in two's complement notation). To use the external format, specify INTEGER EXTERNAL.

If the source data type is INTEGER, DECIMAL, FLOAT, BIGINT, or DECFLOAT (either 4-byte or 8-byte format), an error occurs when the data is greater than 32,767 or less than -32,768.

A SMALLINT output field requires 2 bytes, and the length option is not available.

INTEGER
Specifies that the output field is a 4-byte binary integer (a negative number is in two's complement notation).

If the original data type is DECIMAL, FLOAT, BIGINT, or DECFLOAT (either 4-byte or 8-byte format), an error occurs when the original data is greater than 2,147,483,647 or less than -2,147,483,648.

An INTEGER output field requires 4 bytes, and the length option is not available.

INTEGER EXTERNAL
Specifies that the output field is to contain a character string that represents an integer number.

(length)
Indicates the size of the output data in bytes, including a space for the sign character. When the length is given and the character notation does not fit in the space, an error occurs. The default is 20 characters (including a space for the sign).

If the value is negative, a minus sign precedes the numeric digits. If the output field size is larger than the length of the data, the output data is left justified and blanks are padded on the right.

If the source data type is DECIMAL, FLOAT (either 4-byte or 8-byte format), or DECFLOAT (either 8-byte or 16-byte format), an error occurs when the original data is greater than 9,223,372,036,854,775,807 or less than -9,223,372,036,854,775,808.

BIGINT
Specifies that the output field is an 8-byte binary integer (a negative number is in two's complement notation). To use the external format, specify INTEGER EXTERNAL.

If the original data type is DECIMAL, FLOAT, or DECFLOAT (either 4-byte or 8-byte format), an error occurs when the original data is greater than 9,223,372,036,854,775,807 or less than -9,223,372,036,854,775,808.

BINARY(length)
Indicates that the output field is a binary string type with a fixed length. If the source table column can be null, a NULL indicator byte is placed at the
beginning of the output field for a non-delimited output file. No data conversion is applied to the field. The default for X'\text{strip-char}' is hexadecimal zero (X'00').

**TRUNCATE**
Indicates that the output binary string (encoded for output) is to be truncated from the right, if the data does not fit in the available space for the field in the output records. Without TRUNCATE, an error occurs when the output field size is too small for the data.

**VARBINARY**
Indicates that the output field is a binary string type with varying length. A 2-byte binary field indicating the length of data in bytes is prepended to the data field. If the table column can be null, a NULL indicator byte is placed before the length field for a non-delimited output file. No data conversion is applied to the field. The default for X'\text{strip-char}' is hexadecimal zero (X'00').

**STRIP**
Specifies that UNLOAD is to remove binary zeroes (the default) or the specified string from the beginning, the end, or both ends of the data. UNLOAD adjusts the VARBINARY length field (for the output field) to the length of the stripped data.

**BOTH**
Indicates that UNLOAD is to remove occurrences of binary zeroes or the specified strip character from the beginning and end of the data.
The default is **BOTH**.

**TRAILING**
Indicates that UNLOAD is to remove occurrences of binary zeroes or the specified strip character from the end of the data.

**LEADING**
Indicates that UNLOAD is to remove occurrences of binary zeroes or the specified strip character from the beginning of the data.

X'\text{strip-char}'
Specifies a single-byte character that is to be stripped. It can be specified only in the hexadecimal form, X'\text{hex-string}', where \text{hex-string} is two hexadecimal characters that represent a single SBCS character.

**TRUNCATE**
Indicates that a binary string (encoded for output) is to be truncated from the right, if the data does not fit in the available space for the field in the output records. Without TRUNCATE, an error occurs when the output field size is too small for the data.

**DECIMAL**
Specifies that the output data is a number that is represented by the indicated decimal format (either PACKED, ZONED, or EXTERNAL). If you specify the keyword DECIMAL by itself, packed-decimal format is assumed.

**PACKED**
Specifies that the output data is a number that is represented by the packed-decimal format. You can use DEC or DEC PACKED as an abbreviated form of the keyword.

The packed-decimal representation of a number is of the form \text{ddd...ds}, where \text{d} is a decimal digit that is represented by 4 bits, and \text{s} is a 4-bit sign character (hexadecimal A, C, E, or F for a positive number, and hexadecimal B or D for a negative number).
**length**
Specifies the number of digits (not including the sign digit) that are to be placed in the output field. The length must be between 1 and 31. If the length is odd, the size of the output data field is \((\text{length}+1)/2\) bytes; if even, \((\text{length}/2)+1\) byte.

If the source data type is DECIMAL and the **length** parameter is omitted, the default length is determined by the column attribute defined on the table. Otherwise, the default length is 31 digits (16 bytes).

**scale**
Specifies the number of digits to the right of the decimal point. (Note that, in this case, a decimal point is not included in the output field.) The number must be an integer that is greater than or equal to zero and less than or equal to the length.

The default depends on the column attribute that is defined on the table. If the source data type is DECIMAL, the defined **scale** value is the default value; otherwise, the default value is 0.

If you specify the output field size as less than the length of the data, an error occurs. If the specified field size is greater than the length of data, X'0' is padded on the left.

**ZONED**
Specifies that the output data is a number that is represented by the zoned-decimal format. You can use DEC ZONED as an abbreviated form of the keyword.

The zoned-decimal representation of a number is of the form \(znznzn...zn/zn\), where \(n\) denotes a 4 bit decimal digit (called the numeric bits); \(z\) is the digit’s zone (left 4 bits of a byte); \(s\) is the right-most operand that can be a zone (\(z\)) or can be a sign value (hexadecimal A, C, E, or F for a positive number, and hexadecimal B or D for a negative number).

**length**
Specifies the number of bytes (that is the number of decimal digits) that are placed in the output field. The length must be between 1 and 31.

If the source data type is DECIMAL and the **length** parameter is omitted, the default length is determined by the column attribute that is defined on the table. Otherwise, the default length is 31 bytes.

**scale**
Specifies the number of digits to the right of the decimal point. (Note that, in this case, a decimal point is not included in the output field.) The number must be an integer greater than or equal to zero and less than or equal to the length.

The default depends on the column attribute that is defined on the table. If the source data type is DECIMAL, the defined **scale** value is the default value; otherwise, the default value is 0.

If you specify the output field size as less than the length of the data, an error occurs. If the specified field size is greater than the length of data, X'F0' is padded on the left.
EXTERNAL
Specifies that the output data is a character string that represents a number in the form of ±dd...d.ddd...d, where d is a numeric character 0-9. (The plus sign for a positive value is omitted.)

length
Specifies the overall length of the output data (the number of characters including a sign, and a decimal point if scale is specified).

If the source data type is DECIMAL and the length parameter is omitted, the default length is determined by the column attribute that is defined on the table. Otherwise, the default length is 33 (31 numeric digits, plus a sign and a decimal point). The minimum value of length is 3 to accommodate the sign, one digit, and the decimal point.

scale
Specifies the number of digits to the right of the decimal point. The number must be an integer that is greater than or equal to zero and less than or equal to length - 2 (to allow for the sign character and the decimal point).

If the source data type is DECIMAL and the length parameter is omitted, the default scale is determined by the column attribute that is defined on the table. Otherwise, the default value is 0.

An error occurs if the character representation of a value does not fit in the given or default field size (precision). If the source data type is floating point and a data item is too small for the precision that is defined by scale, the value of zero (not an error) is returned.

FLOAT(length)
Specifies that the output data is a binary floating-point number (32-bit or single-precision FLOAT if the length is between one and 21 inclusive; 64-bit or double-precision FLOAT if the length is between 22 and 53 inclusive). If the length parameter is omitted, the 64-bit format is assumed (output field size is 8 bytes). Note that the length parameter for the FLOAT type does not represent the field size in bytes.

The format of the binary floating-point output is controlled by the global FLOAT option. The default is S/390 format (Hexadecimal Floating Point or HFP). If you specify FLOAT(IEEE), all the binary floating-point output is in IEEE format (Binary Floating Point or BFP). When you specify FLOAT(IEEE) and the source data type DOUBLE is unloaded as REAL, an error occurs if the source data cannot be expressed by the IEEE (BFP) 32-bit notation.

EXTERNAL(length)
Specifies that the output data is a number that is represented by a character string in floating-point notation, ±d.dd...dddE±nn, where d is a numeric character (0-9) for the significant digits; nn after the character E, and the sign consists of two numeric characters for the exponent.

(length)
Specifies the total field length in bytes, including the first sign character, the decimal point, the E character, the second sign character, and the two-digit exponent. If the number of characters in the result is less than the specified or the default length, the result is padded to the right with blanks. The length, if specified, must be greater than or equal to 8.

The default output field size is 14 if the source data type is the 32-bit FLOAT; otherwise, the default is 24.
A FLOAT EXTERNAL output field requires a space of at least seven characters in the output record to accommodate the minimal floating point notation. Otherwise, an error occurs.

**DOUBLE**
Specifies that the output data is in 64-bit floating point notation. If DOUBLE is used, the `length` parameter must not be specified.

**REAL**
Specifies that the output data is in 32-bit floating point notation. If REAL is used, the `length` parameter must not be specified.

**DATE EXTERNAL**
Specifies that the output field is for a character string representation of a date. The output format of date depends on the Db2 installation.

(length)
Specifies the size of the data field in bytes in the output record. A DATE EXTERNAL field requires a space of at least 10 characters. If the space is not available, an error occurs. If the specified `length` is larger than the size of the data, blanks are padded on the right.

**TIME EXTERNAL**
Specifies that the output field is for a character string representation of a time. The output format of time depends on the Db2 installation.

(length)
Specifies the size of the data field in bytes in the output record. A TIME EXTERNAL field requires a space of at least eight characters. If the space is not available, a conversion error occurs. If the specified `length` is larger than the size of the data, blanks are padded on the right.

**TIMESTAMP EXTERNAL**
Specifies that the output field is for a character string representation of a timestamp.

(length)
Specifies the size of the data field in bytes in the output record. A TIMESTAMP EXTERNAL field requires a space of at least 19 characters. If the space is not available, an error occurs. The `length` parameter, if specified, determines the output format of the TIMESTAMP. If the specified `length` is larger than the size of the data, the field is padded on the right with the default padding character.

**TIMESTAMP WITH TIME ZONE EXTERNAL**
Specifies that the output field is for a character string representation of a timestamp.

(length)
Specifies the size of the data field in bytes in the output record. A TIMESTAMP WITH TIME ZONE EXTERNAL field requires a space of at least 26 characters. If the space is not available, an error occurs. The length parameter, if specified, determines the output format of the TIMESTAMP WITH TIME ZONE. If the specified `length` is larger than the size of the data, the field is padded on the right with the default padding character.

**CONSTANT**
Specifies that the output records are to have an extra field containing a constant value. The field name that is associated with the CONSTANT keyword must not coincide with a table column name (the field name is for
clarification purposes only). A CONSTANT field always has a fixed length that is equal to the length of the given string.

'string'
Specifies the character string that is to be inserted in the output records at the specified or default position. A string is the required operand of the CONSTANT option. If the given string is in the form 'string', it is assumed to be an EBCDIC SBCS string. However, the output string for a CONSTANT field is in the specified or default encoding scheme. (That is, if the encoding scheme used for output is not EBCDIC, the SBCS CCSID conversion is applied to the given string before it is placed in output records.)

X'hex-string'
Specifies the character string in hexadecimal form, X'hex-string', that is to be inserted in the output records at the specified or default position. If you want to specify a CONSTANT string value in an encoding scheme other than SBCS EBCDIC, use the hexadecimal form. No CCSID conversion is performed if the hexadecimal form is used.

For a CONSTANT field, no other field selection list options should be specified.

If a CONSTANT field is inserted, it will not be included in the generated LOAD statement (the LOAD statement is generated so that the CONSTANT field is skipped).

If you specify both FORMAT DELIMITED and CONSTANT, the generated LOAD statement is not usable.

ROWID
Specifies that the output data is of type ROWID. The field type ROWID can be specified if and only if the column that is to be unloaded is of type ROWID. The keyword is provided for consistency purposes.

ROWID fields have varying length and a 2-byte binary length field is prepended to the actual data field.

For the ROWID type, no data conversion nor truncation is applied. If the output field size is too small to unload ROWID data, an error occurs.

If the source is an image copy and a ROWID column is selected, and if the page set header page is missing in the specified data set, the UNLOAD utility terminates with the error message DSNU1228I. This situation can occur when the source is an image copy data set of DSNUM that is greater than one for a nonpartitioned table space that is defined on multiple data sets.

BLOB
Indicates that the column is to be unloaded as a binary large object (BLOB). No data conversion is applied to the field.

When you specify the BLOB field type, a 4-byte binary length field is placed in the output record prior to the actual data field. If the source table column can be null, a NULL indicator byte is placed before the length field.

(length)
Specifies the maximum length of the actual data field in bytes. If you specify NOPAD, it indicates the maximum allowable space for the data in the output records; otherwise, the space of the specified length is reserved for the data.

The maximum allowable value is 32767.
The default is the maximum length that is defined on the source table column.

**TRUNCATE**

Indicates that a BLOB string is to be truncated from the right, if the data does not fit in the available space for the field in the output record. For BLOB data, truncation occurs at a byte boundary. Without TRUNCATE, an error occurs when the output field size is too small for the data.

**CLOB**

Indicates that the column is to be unloaded as a character large object (CLOB).

When you specify the CLOB field type, a 4-byte binary length field is placed in the output record prior to the actual data field. If the source table column can be null, a NULL indicator byte is placed before the length field.

If you specify the EBCDIC, ASCII, UNICODE, or CCSID options, the output data is encoded in the CCSID corresponding to the specified option, depending on the subtype of the source data (SBCS or MIXED). No conversion is applied if the subtype is BIT.

(length)

Specifies the maximum length of the actual data field in bytes. If you specify NOPAD, it indicates the maximum allowable space for the data in the output records; otherwise, the space of the specified length is reserved for the data.

The maximum allowable value is 32767.

The default is the maximum length that is defined on the source table column.

**TRUNCATE**

Indicates that a CLOB string (encoded for output) is to be truncated from the right, if the data does not fit in the available space for the field in the output record. For CLOB data, truncation occurs at a character boundary. See “Specifying TRUNCATE and STRIP options for output data” on page 879 for the truncation rules that are used in the UNLOAD utility. Without TRUNCATE, an error occurs when the output field size is too small for the data.

**DBCLOB**

Indicates that the column is to be unloaded as a double-byte character large object (DBCLOB).

If you specify the DBCLOB field type, a 4-byte binary length field is placed in the output record prior to the actual data field. If the source table column can be null, a NULL indicator byte is placed before the length field.

If you specify the EBCDIC, ASCII, UNICODE, or CCSID options, the output data is encoded in the CCSID corresponding to the specified option; DBCS CCSID is used.

(length)

Specifies the maximum length of the actual data field in the number of DBCS characters. If you specify NOPAD, it indicates the maximum allowable space for the data in the output records; otherwise, the space of the specified length is reserved for the data.

The maximum allowable value is 32767.

The default is the maximum length that is defined on the source table column.
TRUNCATE
Indicates that a DBCS string (encoded for output) is to be truncated from the right, if the data does not fit in the available space for the field in the output record. For a DBCLOB data, truncation occurs at a character (DBCS) boundary. See "Specifying TRUNCATE and STRIP options for output data" on page 879 for the truncation rules that are used in the UNLOAD utility. Without TRUNCATE, an error occurs when the output field size is too small for the data.

DECFLOAT (length)
Specifies either a 128-bit decimal floating-point number or a 64-bit decimal floating-point number. The value of the length must be either 16 or 34. If the length is 16, the number is in 64 bit decimal floating-point number format. If the length is 34, the number is in 128 bit decimal floating-point format. The default length is determined by the column attribute defined on the table. Otherwise, the default length is 34 (16 bytes).

DECFLOAT EXTERNAL
Specifies a string of characters that represent a number. The format is an SQL numeric constant.

(length)
Specifies the total field length in bytes. This length includes the first sign character, the decimal point, the E character, the second sign character, and the exponent if in the string. If the number of characters in the result is less than the specified or the default length, the result is padded to the right with blanks. The character representation of a value must fit in the given or default field size.

The default output field size is 23 if the source data type is the DECFLOAT(16). Otherwise, the default is 42.

XML
Specifies that an XML column is being unloaded directly to the output record.

BINARYXML Specifies that the XML document is to be unloaded in binary XML format.

WHEN
Indicates which records in the table space are to be unloaded. If no WHEN clause is specified for a table in the table space, all of the records are unloaded.

The option following WHEN describes the conditions for unloading records from a table.

Data in the table can be in EBCDIC, ASCII, or Unicode. If the target table is in Unicode and the character constants are specified in the utility control statement as EBCDIC, the UNLOAD utility converts these constants to Unicode. To use a constant when the target table is ASCII, specify the hexadecimal form of the constant (instead of the character string form) in the condition for the WHEN clause.

selection condition
Specifies a condition that is true, false, or unknown about a given row. When the condition is true, the row qualifies for UNLOAD. When the condition is false or unknown, the row does not qualify.

The result of a selection condition is derived by application of the specified logical operators (AND and OR) to the result of each specified predicate. If logical operators are not specified, the result of the selection condition is the result of the specified predicate.
Selection conditions within parentheses are evaluated first. If the order of evaluation is not specified by parentheses, AND is applied before OR.

If the control statement is in the same encoding scheme as the input data, you can code character constants in the control statement. Otherwise, if the control statement is not in the same encoding scheme as the input data, you must code the condition with hexadecimal constants. For example, if the table space is in EBCDIC and the control statement is in UTF-8, use \(1:1 = X'31'\) in the condition rather than \(1:1 = '1'\).

**Restriction**: UNLOAD cannot filter rows that contain encrypted data.

**predicate**

Specifies a condition that is true, false, or unknown about a row.

In the predicate, you cannot specify a DECFLOAT constant or a column of any of the following types:

- DECFLOAT
- LONG VARCHAR
- LONG VARGRAPHIC
- ROWID
- CLOB
- BLOB
- DBCLOB

You can specify an XML column only with IS NULL or IS NOT NULL.

You can specify a VARCHAR column only with the following predicates:

- IS NULL
- IS NOT NULL
- A comparison with a HEX or fixed-length CHAR value

You can specify a VARGRAPHIC column only with the following predicates:

- IS NULL
- IS NOT NULL
- A comparison with a HEX or fixed-length GRAPHIC value

Column names in the predicate are case-sensitive. For example, if a column in the source table is named SALARY, SALARY=20000 is a valid predicate, but salary=20000 is not a valid predicate.

**basic predicate**

Specifies the comparison of a column with a constant. If the value of the column is null, the result of the predicate is unknown. Otherwise, the result of the predicate is true or false.

**column = constant**

The column is equal to the constant or labeled duration expression.

**column < > constant**

The column is not equal to the constant or labeled duration expression.

**column > constant**

The column is greater than the constant or labeled duration expression.
column < constant
The column is less than the constant or labeled duration expression.

column >= constant
The column is greater than or equal to the constant or labeled duration expression.

column <= constant
The column is less than or equal to the constant or labeled duration expression.

Note: The following alternative comparison operators are available:
!= or ^= for not equal.
>` or => for not greater than.
!< or <^ for not less than.

The symbol ¬ representing “not” is supported for compatibility purposes. Use ! where possible.

**BETWEEN predicate**
Indicates whether a given value is between two other given values that are specified in ascending order. The values can be constants or labeled duration expressions. Each of the predicate’s two forms (BETWEEN and NOT BETWEEN) has an equivalent search condition, as shown in the following table. When relevant, the table also shows any equivalent predicates.

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Equivalent predicate</th>
<th>Equivalent search condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>column BETWEEN value1</td>
<td>None</td>
<td>(column &gt;= value1 AND column &lt;= value2)</td>
</tr>
<tr>
<td>AND value2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>column NOT BETWEEN value1</td>
<td>NOT(column BETWEEN</td>
<td>(column &lt; value1 OR column &gt; value2)</td>
</tr>
<tr>
<td>AND value2</td>
<td>value1 AND value2)</td>
<td></td>
</tr>
</tbody>
</table>

Note: The values can be constants or labeled duration expressions.

For example, the following predicate is true for any row when salary is greater than or equal 10000 and less than or equal to 20000:

\[
\text{SALARY BETWEEN 10000 AND 20000}
\]

**IN predicate**
Specifies that a value is to be compared with a set of values. In the IN predicate, the second operand is a set of one or more values that are specified by constants. Each of the predicate’s two forms (IN and NOT IN) has an equivalent search condition, as shown in the following table.

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Equivalent search condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>value1 IN (value1, value2,..., valuen)</td>
<td>(value1 = value2 OR ... OR value1 = valuen)</td>
</tr>
<tr>
<td>value1 NOT IN (value1, value2,..., valuen)</td>
<td>value1 = value2 AND ... AND value1 = valuen)</td>
</tr>
</tbody>
</table>

Note: The values can be constants.

For example, the following predicate is true for any row whose employee is in department D11, B01, or C01:

\[
\text{WORKDEPT IN (‘D11’, ‘B01’, ‘C01’)}
\]
LIKE predicate

Specifies the qualification of strings that have a certain pattern.

Within the pattern, a percent sign or underscore can have a special meaning, or it can represent the literal occurrence of a percent sign or underscore. To have its literal meaning, it must be preceded by an escape character. If it is not preceded by an escape character, it has its special meaning. The underscore character (_) represents a single, arbitrary character. The percent sign (%) represents a string of zero or more arbitrary characters.

The ESCAPE clause designates a single character. That character, and only that character, can be used multiple times within the pattern as an escape character. When the ESCAPE clause is omitted, no character serves as an escape character, so that percent signs and underscores in the pattern always have their special meanings.

The following rules apply to the use of the ESCAPE clause:

- The ESCAPE clause cannot be used if x is mixed data.
- If x is a character string, the data type of the string constant must be character string. If x is a graphic string, the data type of the string constant must be graphic string. In both cases, the length of the string constant must be 1.
- The pattern must not contain the escape character except when followed by the escape character, '%', or '_'. For example, if '+' is the escape character, any occurrence of '+' other than '++', '+_', or '+%' in the pattern is an error.

When the pattern does not include escape characters, a simple description of its meaning is:

- The underscore sign (_) represents a single arbitrary character.
- The percent sign (%) represents a string of zero or more arbitrary characters.
- Any other character represents a single occurrence of itself.

Let x denote the column that is to be tested and y the pattern in the string constant. The following rules apply to predicates of the form "x LIKE y...". If NOT is specified, the result is reversed.

- When x and y are both neither empty nor null, the result of the predicate is true if x matches the pattern in y and false if x does not match the pattern in y.
- When x or y is null, the result of the predicate is unknown.
- When y is empty and x is not empty, the result of the predicate is false.
- When x is empty and y is not empty, the result of the predicate is false unless y consists only of one or more percent signs.
- When x and y are both empty, the result of the predicate is true.

The pattern string and the string that is to be tested must be of the same type. That is, both x and y must be character strings, or both x and y must be graphic strings. When x and y are graphic strings, a character is a DBCS character. When x and y are character strings and x is not mixed data, a character is an SBCS character and y is interpreted as SBCS data regardless of its subtype.

Strings and patterns
The string \( y \) is interpreted as a sequence of the minimum number of substring specifiers such that each character of \( y \) is part of exactly one substring specifier. A substring specifier is an underscore, a percent sign, or any non-empty sequence of characters other than an underscore or percent sign.

The string \( x \) matches the pattern \( y \) if a partitioning of \( x \) into substrings exists, such that:

- A substring of \( x \) is a sequence of zero or more contiguous characters, and each character of \( x \) is part of exactly one substring.
- If the \( n \)th substring specifier is an underscore, the \( n \)th substring of \( x \) is any single character.
- If the \( n \)th substring specifier is a percent sign, the \( n \)th substring of \( x \) is any sequence of zero or more characters.
- If the \( n \)th substring specifier is neither an underscore nor a percent sign, the \( n \)th substring of \( x \) is equal to that substring specifier and has the same length as that substring specifier.
- The number of substrings of \( x \) is the same as the number of substring specifiers.

When escape characters are present in the pattern string, an underscore, percent sign, or escape character represents a single occurrence of itself if and only if it is preceded by an odd number of successive escape characters.

The way a pattern is matched to evaluate the LIKE predicate depends on whether blanks at the end of fixed length strings are significant, or if the blanks are ignored. When the LIKE_BLANK_IN_SIGNIFICANT subsystem parameter is enabled, the LIKE predicate can produce different results.

**Mixed data patterns:** If \( x \) is mixed data, the pattern is assumed to be mixed data, and its special characters are interpreted as follows:

- A single-byte underscore refers to one single-byte character; a double-byte underscore refers to one double-byte character.
- A percent sign, either single-byte or double-byte, refers to any number of characters of any type, either single-byte or double-byte.
- Redundant shift bytes in \( x \) or \( y \) are ignored.

**Related information:**

[LIKE predicate (Db2 SQL)]

**NULL predicate**

Specifies a test for null values.

If the value of the column is null, the result is true. If the value is not null, the result is false. If NOT is specified, the result is reversed. (That is, if the value is null, the result is false, and if the value is not null, the result is true.)

**labeled duration expression**

Specifies an expression that begins with special register CURRENT DATE or special register CURRENT TIMESTAMP (the forms CURRENT_DATE and CURRENT_TIMESTAMP are also acceptable). For CURRENT_TIMESTAMP, if the comparison is with a timestamp column, the timestamp precision of the special register will be the same as the column timestamp precision. Otherwise default timestamp precision will be used. This special register can be followed by arithmetic operations of addition.
or subtraction. These operations are expressed by using numbers that are followed by one of the seven duration keywords: YEARS, MONTHS, DAYS, HOURS, MINUTES, SECONDS, or MICROSECONDS. (The singular form of these keywords is also acceptable: YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, and MICROSECOND.)

Utilities always evaluate a labeled duration expression as a timestamp and implicitly convert to a date if the comparison is with a date column.

**Incrementing and decrementing CURRENT DATE:** The result of adding a duration to a date, or of subtracting a duration from a date, is itself a date. (For the purposes of this operation, a month denotes the equivalent of a calendar page. Adding months to a date, then, is like turning the pages of a calendar, starting with the page on which the date appears.) The result must fall between the dates January 1, 0001 and December 31, 9999 inclusive. If a duration of years is added or subtracted, only the year portion of the date is affected. The month is unchanged, as is the day, unless the result would be February 29 of a non-leap-year. In this situation, the day portion of the result is set to 28.

Similarly, if a duration of months is added or subtracted, only months and, if necessary, years are affected. The day portion of the date is unchanged unless the result would be invalid (September 31, for example). In this case the day is set to the last day of the month.

Adding or subtracting a duration of days affects the day portion of the date, and potentially the month and year.

Date durations, whether positive or negative, can also be added to and subtracted from dates. As with labeled durations, the result is a valid date.

When a positive date duration is added to a date, or a negative date duration is subtracted from a date, the date is incremented by the specified number of years, months, and days.

When a positive date duration is subtracted from a date, or a negative date duration is added to a date, the date is decremented by the specified number of days, months, and years.

Adding a month to a date gives the same day one month later, unless that day does not exist in the later month. In that case, the day in the result is set to the last day of the later month. For example, January 28 plus one month gives February 28; one month added to January 29, 30, or 31 results in either February 28 or, for a leap year, February 29. If one or more months is added to a given date and then the same number of months is subtracted from the result, the final date is not necessarily the same as the original date.

The order in which labeled date durations are added to and subtracted from dates can affect the results. When you add labeled date durations to a date, specify them in the order of YEARS + MONTHS + DAYS. When you subtract labeled date durations from a date, specify them in the order of DAYS - MONTHS - YEARS. For example, to add one year and one day to a date, specify the following code:

When the labeled duration expression begins with special register CURRENT_TIMESTAMP, the CURRENT_TIMESTAMP is compared with the time zone column. The timestamp precision of the special register will be the same as the column timestamp precision. Otherwise the default timestamp precision will be used. The time zone of CURRENT
TIMESTAMP is the value of special register CURRENT TIMEZONE. The comparison is done by comparing the UTC portion.

CURRENT DATE + 1 YEAR + 1 DAY

To subtract one year, one month, and one day from a date, specify the following code:

CURRENT DATE - 1 DAY - 1 MONTH - 1 YEAR

Incrementing and decrementing timestamps: The result of adding a duration to a timestamp, or of subtracting a duration from a timestamp, is itself a timestamp. Date and time arithmetic is performed as previously defined, except that an overflow or underflow of hours is carried into the date part of the result, which must be within the range of valid dates.

Related reference:
- EDITPROCs and VALIDPROCs for handling basic and reordered row formats (Db2 Administration Guide)
- CREATE TABLE (Db2 SQL)

Related information:
- Converting basic row format table spaces with edit and validation routines to reordered row format (Db2 Administration Guide)

### Unloading partitions

You can unload partitions in one of two ways.

**About this task**

Regardless of the method, the unloaded data can be stored in a single data set for all selected partitions or in one data set for each selected partition. If you want to unload to a single output data set, specify a DD name to UNLDDN. If you want to unload into multiple output data sets, specify a template name that is associated with the partitions. You can process multiple partitions in parallel if the TEMPLATE definition contains the partition as a variable, for example &PA.

You cannot specify multiple output data sets with the FROMCOPY or the FROMCOPYDDN option.

**Procedure**

If the source table space is partitioned, use only one of the following methods to select the partitions to unload:

- Use the LIST keyword with a LISTDEF that contains PARTLEVEL specifications. Partitions can be either included or excluded by the use of the INCLUDE and the EXCLUDE features of LISTDEF.
- Specify the PART keyword to select a single partition or a range of partitions.

### Unloading XML data

You can unload XML data in one of two ways.

**About this task**

XML columns can be unloaded with either of the following methods:
• The XML column can be unloaded to the output records. XML column value can be placed in the OUTPUT record with or without any other unloading column values. The output record can be in delimited or non-delimited format. For a non-delimited format, the XML column is handled like a variable character with a 2-byte length preceding the XML value. For a delimited format there are no length bytes present. If the total output record length is more than 32 KB, unload the record in spanned record format by specifying the SPANNED YES option.

• The XML column can be unloaded to a separate file whether the XML column length is less than 32K or not.

To unload XML data directly to output record:

Specify XML as the output field type. If the output is a non-delimited format, a 2-byte length will precede the value of the XML. For delimited output, no length field is present. XML is the only acceptable field type when unloading the XML directly to the output record. No data type conversion applies and you cannot specify FROMCOPY.

If the input data is in Extensible Dynamic Binary XML Db2 Client/Server Binary XML Format (binary XML format), you need to specify BLOBF BINARYXML.

To unload XML data to a separate file:

• In the UNLOAD utility control statement, specify BLOBF, CLOBF or DBCLOBF. These keywords indicate that the output column contains the name of a file to which the XML value is to be unloaded. Also specify either CHAR or VARCHAR instead of XML. Do not specify FROMCOPY.

Example: The following UNLOAD statement specifies that the data from the XML column ORDER_CREATE_XML1 is to be unloaded to the file that is defined by template BLOBFC1.

```sql
UNLOAD DATA FROM TABLE SCQA0000.TB_ORDER_PBR
  (ORDER_CREATE_XML1 POSITION(+) VARCHAR BLOBF BLOBFC1,
   ORDER_ALL_LOCAL POSITION(+) INTEGER )
```

• Use the template control statement to create the XML output file and filename. If data sets are not created and the DSN type is not specified on the template, UNLOAD will use PDS as the data set type. PDS has a limit of single volume. The output file uses multiple volumes, so you must specify HFS as the DSN type.

---

**Unloading LOB data**

You can unload LOB data in one of two ways.

**About this task**

LOB columns can be unloaded with either of the following methods:

• The LOB column can be unloaded to the output records. The LOB column value can be placed in the OUTPUT record with or without any other unloading column values. The output record can be in delimited or non-delimited format. For a non-delimited format, the LOB column is handled like a variable character with a 2-byte length preceding the LOB value. For a delimited format there are
no length bytes present. If the total output record length is more than 32 KB, unload the record in spanned record format by specifying the SPANNED YES option.

- The LOB column can be unloaded to a separate file whether the LOB column length is less than 32K or not.

**Procedure**

To unload LOB data, use one of the following methods:

- To unload LOB data directly to output record:
  Specify LOB as the output field type. If the output is a non-delimited format, a 2-byte length will precede the value of the LOB. For delimited output, no length field is present. LOB is the only acceptable field type when unloading the LOB data directly to the output record. No data type conversion applies and you cannot specify FROMCOPY.

- To unload LOB data to a separate file:
  - Create an UNLOAD utility control statement. Specify BLOBF, CLOBF or DBCLOBF to indicate that the output column contains a filename which the LOB value is to be unloaded. You cannot specify FROMCOPY.
  - Use the template control statement to create the LOB output file and filename. If data sets are not created and the DSN type is not specified on the template, UNLOAD will use PDS as the data set type. PDS has a limit of single volume. The output file uses multiple volumes, so you must specify HFS as the DSN type.

If you unload data to a separate file, and the LOB column from which you unload is empty, the data set that is specified by UNLDDN contains one of the following items:

- A blank file name if the source column is specified as CHAR CLOBF, CHAR BLOBF, or CHAR DBCLOBF
- A file name with length 0 if the source column is specified as VARCHAR CLOBF, VARCHAR BLOBF, or VARCHAR DBCLOBF

The UNLOAD utility does not create a data set or file for the empty LOB.

---

**Unloading data in spanned record format**

If you want to unload data from a table that has large LOB or XML fields, consider unloading the data in spanned record format to improve the performance of read/write operations.

**About this task**

When you unload data in spanned record format, all LOB and XML data for a table space or table space partition can be written to an individual sequential file. This file can reside on DASD and can span multiple volumes. Having such a single sequential file can improve the performance of read/write operations.

UNLOAD SPANNED YES ignores large block interface (LBI) if used by any output data sets.

**Procedure**

To unload data in spanned record format:
Specify the SPANNED YES option. Specify in the field specification list that all LOB and XML data are to be at the end of the record.

**Example:** The following UNLOAD statement specifies that the data from table TB1 is to be unloaded in spanned record format. Notice that in the field specification list, the CLOB columns are listed at the end and POSITION is not specified.

```
UNLOAD TABLESPACETESTDB1.CLOBBASE SPANNED YES
  FROM TABLE TB1
    (ID,
     C1 INTEGER,
     C2 INTEGER,
     C3 CHAR(100),
     C4 CHAR(100),
     C5 INTEGER,
     C6 CHAR(100),
     C7 CHAR(100),
     C8 CHAR(100),
     CLOB1 CLOB,
     CLOB2 CLOB,
     CLOB3 CLOB)
```

**Results**

**Example of spanned record format:** The following figure shows a conceptual example of a spanned record that was unloaded.

```
<table>
<thead>
<tr>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>C6</th>
<th>C7</th>
<th>C8</th>
<th>Start of CLOB 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The rest of CLOB 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Start of CLOB 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>more of CLOB 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>the rest of CLOB 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CLOB 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The next row.........</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>------------------</td>
</tr>
</tbody>
</table>
```

**What to do next**

When you run LOAD on data that was unloaded in spanned record format, use the LOAD statements that are in the SYSPUNCH data sets after UNLOAD runs. Those LOAD statements include SORTKEYS parameters with accurate values. During LOAD, Db2 cannot estimate the size of the sort work data sets by checking the contents of the SYSREC data sets that are produced during UNLOAD with SPANNED YES.

**Related concepts:**

[Large Block Interface (LBI) (z/OS DFSMS Using Data Sets)]

**Selecting tables and rows to unload**

If a table space contains multiple tables, you can select specific tables to unload.

**About this task**

To select tables and rows to unload:
Procedure

In the UNLOAD utility control statement, use the FROM TABLE specification clause. Use one instance of the FROM TABLE clause for each table that is to be unloaded.

Within a FROM TABLE clause, you can specify one or more of the following criteria:

- Row and column selection criteria by using the field specification list
- Row selection conditions by using the WHEN specification clause
- Row sampling specifications

If you do not specify at least one FROM TABLE clause, the rows from all the tables in the table space are unloaded.

If you specify one or more FROM TABLE clauses for a table space, only the qualified rows from the specified tables are unloaded.

Related information:

“FROM-TABLE-spec syntax diagram and option descriptions” on page 838

Selecting and ordering columns to unload

Use a field specification list in a FROM TABLE clause to unload specified columns in the listed order. If you omit a field specification list, all the columns in the row are unloaded in the order of the columns that are defined on the table.

About this task

You can specify a format conversion option for each field in the field specification list.

If you select a LOB column in a list of field specifications or select a LOB column by default (by omitting a list of field specifications), LOB data is materialized in the output. However, you cannot select LOB columns from image copy data sets.

Unloading data from image copy data sets

In addition to unloading data from table spaces and partitions, you can also unload data from one or more image copy data sets. The UNLOAD utility accepts full image copies, incremental image copies, and copies of pieces as valid input sources.

Before you begin

Ensure that the image copy data set that you want to unload from meets the following requirements:

- The source image copy data set must be created by one of the following utilities:
  - COPY
  - COPYTOCOPY
  - LOAD inline image copy
  - MERGECOPY
  - REORG TABLESPACE inline image copy
  - DSN1COPY
- The image copy data set must be for a single table space.
- If you want to unload a ROWID column, the image copy must contain the page set header page.
• If you want to use UNLOAD to process image copies from different versions, the copy must be created with the SYSTEMPAGES YES option.

• If you want to unload compressed records, the image copy can be a full image copy or an incremental image copy. In either case, the copy must be created with the SYSTEMPAGES YES option. If the image copy data set is an incremental image copy or a copy of a partition or partitions, the same data set must contain the dictionary pages for decompression. If an image copy data set contains a compressed row and a dictionary is not available, Db2 issues an error message.

• The copy cannot be a VSAM FlashCopy image copy. If the object does not require system pages in front, you can create an image copy as a workaround. Use the COPY utility or COPYTOCOPY utility to create a sequential format image copy from the FlashCopy image copy. Then, use the sequential format image copy as input for UNLOAD.

• If an image copy contains data for columns that no longer exist in the catalog because the columns were dropped, UNLOAD cannot unload from this image copy. UNLOAD issues message DSNU1227I with return code 8.

About this task

Restriction: You cannot unload LOB data or XML data from copies.

Procedure

To unload data from image copy data sets:

Specify either the FROMCOPY or FROMCOPYDDN option in the UNLOAD utility control statement as follows:

FROMCOPY

Use the FROMCOPY option to unload rows from a single image copy data set.

You can use the FROMCOPY option to specify a full or incremental copy of partitions of a segmented table space that consists of multiple data sets. However, if a mass delete operation occurred for a table in the table space before you created the copy, the utility might not unload the deleted rows. The utility unloads deleted rows only if the space map pages that indicate the mass delete are not included in the data set that corresponds to the specified copy. Therefore, where possible, use the FROMCOPYDDN option to concatenate the copy of table space partitions.

FROMCOPYDDN

Use the FROMCOPYDDN option to unload data from one or more image copy data sets that are associated with the specified DD name.

You can use this option to concatenate the copy of table space partitions under a DD name to form a single input data set image. When you use the FROMCOPYDDN option, concatenate the data sets in the order of the data set number; the first data set must be concatenated first. If the data sets are concatenated in the wrong order or if different generations of image copies are concatenated, the results might be unpredictable. For example, if the most recent image copy data sets and older image copies are intermixed, the results might be unpredictable.

You can also use the FROMCOPYDDN option to concatenate a full image copy and incremental image copies for a table space, a partition, or a piece.
However, duplicate rows are also unloaded. Instead, consider using the MERGECOPY utility to generate an updated full image copy as the input to UNLOAD.

When you specify the FROMCOPY or the FROMCOPYDDN option, you can specify only one output data set. You can select specific rows and columns to unload just as you would for a table space. You can specify the selection criteria with either the PART keyword, the FROM TABLE clause, or both, to qualify tables and rows that are to be unloaded. However, do not include LOB columns in the field specification list. You can unload rows that contain LOB columns only when the LOB columns are excluded. Specify the table space name in the TABLESPACE option. The specified table space must exist when you run UNLOAD. (The table space cannot have been dropped since the image copy was taken.) If an image copy contains rows from dropped tables, UNLOAD ignores these rows.

After you run UNLOAD, the image copy data is unloaded to the output data set. However, certain situations can affect the output as follows:

- Suppose that the image copy contains a table to which ALTER ADD COLUMN was applied after the image copy was taken. In this case, UNLOAD sets the system or user-specified default value for the added column when the data is unloaded from such an image copy.
- If an image copy was created by an inline copy operation, the image copy can contain duplicate pages. If duplicate pages exist, UNLOAD issues a warning message, and all the qualified rows in duplicate pages are unloaded into the output data set.
- If the image copy was taken with the SHRLEVEL CHANGE option specified, rows might be updated or moved. As a result, data that is unloaded from such a copy might contain duplicates of these rows.

The later two situations can be prevented by using an image copy that was taken from a consistent FlashCopy.

**Related reference:**

- “Syntax and options of the COPY control statement” on page 135
- “Syntax and options of the COPYTOCOPY control statement” on page 186
- “Syntax and options of the MERGECOPY control statement” on page 378
- “Syntax and options of the UNLOAD control statement” on page 825
- “Sample UNLOAD control statements” on page 881

### Data type conversion with the UNLOAD utility

You can convert one data type to another compatible data type by using the UNLOAD utility. The source type is used for user-defined distinct types.

For example, you can convert columns of a numeric type (SMALLINT, INTEGER, FLOAT, DOUBLE, REAL, and DECIMAL) from the Db2 internal format to the S/390 or an external format.

When you unload a floating-point type column, you can specify the binary form of the output to either the S/390 format (hexadecimal floating point, or HFP), or the IEEE format (binary floating point, or BFP).

You can also convert a varying-length column to a fixed-length output field, with or without padding characters. In either case, unless you explicitly specify a
fixed-length data type for the field, the data itself is treated as a varying-length data, and a length field is appended to the data.

For certain data types, you can unload data into fields with a smaller length by using the TRUNCATE or STRIP options. In this situation, if a character code conversion is applied, the length of the data in bytes might change due to the code conversion. The truncation operation is applied after the code conversion.

You can perform character code conversion on a character type field, including converting numeric columns to the external format and the CLOB type. Be aware that when you apply a character code conversion for mixed-data fields, the length of the result string in bytes can be shorter or longer than the length of the source string. Character type data is always converted if you specify any of the character code conversion options (EBCDIC, ASCII, UNICODE, or CCSID).

DATE, TIME, or TIMESTAMP column types are always converted into the external formats based on the DATE, TIME, and TIMESTAMP formats of your installation.

**Output field types**

An output field can have a different data type from the one that is defined on a source table column if the data types are compatible. The UNLOAD utility follows the general Db2 rules and conventions for the data type attributes and the compatibility among the data types.

If you specify a data type in the UNLOAD control statement, the field type information is included in the generated LOAD utility statement. For specific data type compatibility information, refer to the following table. These tables show the compatibility of the data type of the source column (input data type) with the data type of the output field (output data type). A Y indicates that the input data type can be converted to the output data type.

The following table shows the compatibility of converting numeric data types.

**Table 125. Compatibility of converting numeric data types**

<table>
<thead>
<tr>
<th>Input data types</th>
<th>SMALLINT</th>
<th>INTEGER (external)</th>
<th>BIGINT</th>
<th>DECIMAL (external)</th>
<th>FLOAT (external)</th>
<th>DOUBLE or REAL</th>
<th>FLOAT/REAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMALLINT</td>
<td>Y</td>
<td>Y^1</td>
<td>Y</td>
<td>Y^1</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>INTEGER</td>
<td>Y^2</td>
<td>Y^1</td>
<td>Y</td>
<td>Y^1</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>BIGINT</td>
<td>Y^2</td>
<td>Y^2</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>Y^2</td>
<td>Y^1, 2</td>
<td>Y</td>
<td>Y^1</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>FLOAT, DOUBLE, or REAL</td>
<td>Y^2</td>
<td>Y^1, 2</td>
<td>Y</td>
<td>Y^1, 2</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>DECFLOAT</td>
<td>Y^2</td>
<td>Y^1, 2</td>
<td>Y</td>
<td>Y^1, 2</td>
<td>N^2</td>
<td>Y^3</td>
<td>Y^3</td>
</tr>
</tbody>
</table>

Note:
1. Subject to the CCSID conversion, if specified (EXTERNAL case).
2. Potential overflow (conversion error).
3. When converting from DECFLOAT(34) to DECFLOAT(16), you might encounter overflow, underflow, subnormal number, or inexact. However, there will be no conversion error.

The following table shows the compatibility of converting character data types.
### Table 126. Compatibility of converting character data types

<table>
<thead>
<tr>
<th>Input data types</th>
<th>BLOB</th>
<th>CHAR</th>
<th>VARCHAR</th>
<th>CLOB</th>
<th>GRAPHIC</th>
<th>GRAPHIC EXTERNAL</th>
<th>VARCHAR</th>
<th>DB-CLOB</th>
<th>BINARY</th>
<th>VAR-BINARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOB</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>CLOB</td>
<td>N</td>
<td>Y(^{1,2})</td>
<td>Y(^{1,2})</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>DBCLOB</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y(^{1,2})</td>
<td>Y(^{1,2,3})</td>
<td>Y(^{1,2})</td>
<td>Y(^{1})</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>CHAR</td>
<td>N</td>
<td>Y(^{1})</td>
<td>Y(^{1})</td>
<td>Y(^{1,4})</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>VARCHAR or LONG VARCHAR</td>
<td>N</td>
<td>Y(^{1,2})</td>
<td>Y(^{1})</td>
<td>Y(^{1,4})</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>GRAPHIC</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y(^{1})</td>
<td>Y(^{1,3})</td>
<td>Y(^{1})</td>
<td>Y(^{1})</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>VAR-GRAHMIC or LONG VARCHAR</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y(^{1,2})</td>
<td>Y(^{1,2,3})</td>
<td>Y(^{1})</td>
<td>Y(^{1})</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>BINARY</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>VARBINARY</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

**Note:**

1. Subject to the CCSID conversion, if specified.
2. Results in an error if the field length is too small for the data unless you specify the TRUNCATE option. Note that a LOB has a 4-byte length field; any other varying-length type has a 2-byte length field.
3. Only in the EBCDIC output mode.
4. Not applicable to BIT subtype data.

The following table shows the compatibility of converting time data types.

### Table 127. Compatibility of converting time data types

<table>
<thead>
<tr>
<th>Input data types</th>
<th>DATE EXTERNAL</th>
<th>TIME EXTERNAL</th>
<th>TIMESTAMP EXTERNAL</th>
<th>TIMESTAMP WITH TIME ZONE EXTERNAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE</td>
<td>Y(^{1})</td>
<td>N</td>
<td>Y(^{1,2})</td>
<td>Y(^{1,2})</td>
</tr>
<tr>
<td>TIME</td>
<td>N</td>
<td>Y(^{1})</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>Y(^{1,3})</td>
<td>Y(^{1,3})</td>
<td>Y(^{1})</td>
<td>Y(^{1,2})</td>
</tr>
<tr>
<td>TIMESTAMP WITH TIME ZONE</td>
<td>Y(^{1,4})</td>
<td>Y(^{1,4})</td>
<td>Y(^{1})</td>
<td></td>
</tr>
</tbody>
</table>

**Note:**

1. Subject to the CCSID conversion, if specified.
2. Zeros in the time portion. IMPLICIT_TZ in time zone portion if the output data type is TIMESTAMP WITH TIME ZONE.
3. DATE or TIME portion of the timestamp.
4. DATE, TIME or TIMESTAMP portion of the timestamp with time zone.

**Related concepts:**

Data types (Db2 SQL)

**Related reference:**

"Syntax and options of the UNLOAD control statement“ on page 825
Output field positioning and size

By default, output data is always placed in an output record in the order of the defined columns over the selected tables. You can choose to specify the order of the output fields by using a list of field specifications.

Use the POSITION option to specify field position in the output records. You can also specify the size of the output data field by using the length parameter for a particular data type. The length parameter must indicate the size of the actual data field. The start parameter of the POSITION option indicates the starting position of a field, including the NULL indicator byte (if the field can be null) and the length field (if the field is varying length).

Using the POSITION parameter, the length parameter, or both can restrict the size of the data field in the output records. Use care when specifying the POSITION and length parameters, especially for nullable fields and varying length fields. If a conflict exists between the length parameter and the size of the field in the output record that is specified by the POSITION parameters, Db2 issues an error message, and the UNLOAD utility terminates. If an error occurs, the count of the number of records in error is incremented. See the description of the MAXERR option of UNLOAD for more information.

If you specify a length parameter for a varying-length field and you also specify the NOPAD option, length indicates the maximum length of data that is to be unloaded. Without the NOPAD option, UNLOAD reserves a space of the given length instead of the maximum data size.

If you explicitly specify start parameters for certain fields, they must be listed in ascending order in the field selection list. Unless you specify HEADER NONE for the table, a fixed-length record header is placed at the beginning of each record for the table, and the start parameter must not overlap the record header area.

The TRUNCATE option is available for certain output field types. For the output field types where the TRUNCATE option is not applicable, enough space must be provided in the output record for each field.

Related concepts:

“Field specification errors” on page 881
“Layout of output fields”
“Specifying TRUNCATE and STRIP options for output data” on page 879

Related reference:

“Syntax and options of the UNLOAD control statement” on page 825

Layout of output fields

The layout of output fields vary based on a variety of factors, including whether the field is varying-length or fixed-length and whether the field is nullable.

Output fields can have the following layouts:

- Fixed-length
- Nullable fixed-length
- Varying-length field
- Varying-length field without the NOPAD option
- Nullable varying-length field with the NOPAD option
• Nullable varying-length field without the NOPAD option

The following figure shows the layout of a fixed-length field that cannot be null. This diagram shows that the data field begins at a specified position, or at the next byte position past the end of the previous data field. The data field then continues for the specified length or the length of the column in the table definition. For GRAPHIC EXTERNAL data, shift-in and shift-out characters are inserted before and after the data.

![Figure 80. Layout of a fixed-length field (NOT NULL)](image)

The following figure shows the layout of a fixed-length field that can be null. This diagram shows that a null indicator byte is stored before the data field, which begins at the specified position or at the next byte position past the end of the previous data field.

![Figure 81. Layout of a nullable fixed-length field](image)

If you are running UNLOAD with the NOPAD option and need to determine the layout of a varying-length field that cannot be null, see the layout diagram in the following figure. The length field begins at the specified position or at the next byte position past the end of the previous data field.
For UNLOAD without the NOPAD option, the layout of a varying-length field that cannot be null is depicted in the following figure.

Figure 82. Layout of a varying-length field (NOT NULL) with the NOPAD option

For UNLOAD with the NOPAD option, the layout of a varying-length field that can be null is depicted in the following figure. The length field begins at the specified position or at the next byte position past the end of the previous data field.

Figure 83. Layout of a varying-length field (NOT NULL) without the NOPAD option
For UNLOAD without the NOPAD option, the layout of a varying-length field that can be null is depicted in the following figure. The length field begins at the specified position or at the next byte position past the end of the previous data field.

![Layout of a nullable varying-length field with the NOPAD option](image1)

**Figure 84. Layout of a nullable varying-length field with the NOPAD option**

For UNLOAD without the NOPAD option, the layout of a varying-length field that can be null is depicted in the following figure. The length field begins at the specified position or at the next byte position past the end of the previous data field.

![Layout of a nullable varying-length field without the NOPAD option](image2)

**Figure 85. Layout of a nullable varying-length field without the NOPAD option**

### Output for special values Infinity, sNaN, or NaN

When you run UNLOAD against a DECFLOAT column that contains the special values Infinity, sNaN, or NaN, the output is in uppercase.

Infinity, sNaN, and NaN are unloaded as INFINITY, SNAN, and NAN, respectively.

### Unloading delimited files

You can use the DELIMITED option to specify that UNLOAD is to produce an output file in delimited format. All fields in the output data set are either in character string or numeric external format. Each column is separated from the next by a column delimiter, and character strings are marked by character string delimiters.
**Recommendation:** If a delimited file is to be transferred to or from a platform other than z/OS or between Db2 for z/OS systems that use different EBCDIC or ASCII CCSIDs, use Unicode as the encoding scheme for the delimited file. Using Unicode avoids possible CCSID translation problems.

You are responsible for ensuring that the chosen delimiters are not part of the data in the file. If the delimiters are part of the file’s data, unexpected errors can occur.

**Restrictions:** The following general restrictions apply to the use of delimiters:

- You cannot specify the same character for more than one type of delimiter (COLDEL, CHARDEL, and DEPT).
- You can specify a character constant for a delimiter if the utility control statement is coded in the same encoding scheme as the output file. For example, the utility control statement is coded in Unicode and the output data is also coded in Unicode.
- Use the hex representation for non-default delimiters if the utility control statement is coded in a different encoding scheme than the output file. For example, the utility control statement is coded in Unicode and the output file is coded in EBCDIC. In this case, if you do not use the hex representation for the non-default delimiters, the results can be unpredictable.
- You cannot specify HEADER OBID and ROWID for output fields in delimited output format. Because a header is not allowed, output must be from a single table.
- When you specify the DELIMITED option, the utility ignores the POSITION keyword. The utility overrides field data type specifications according to the specifications of the delimited format. (For example, length values for CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, CLOB, DBCLOB, and BLOB data are the delimited lengths of each field in the output data set, and the utility unloads all numeric types in external format.)
- You cannot specify a binary 0 (zero) for any delimiter.
- No null byte is present for a delimited output file. A null value is indicated by the absence of a cell value where one would normally occur. For example, two successive column delimiters or a missing column at the end of a record indicate a null value.
- You cannot use the default decimal point as a character string delimiter (CHARDEL) or a column string delimiter (COLDEL).
- Shift-in and shift-out characters cannot be specified as EBCDIC MBCS delimiters.
- In the DBCS environment, the pipe character ( | ) is not supported.
- If the output is coded in ASCII or Unicode, you cannot specify any of the following values for any delimiter: X'0A', X'0D', X'2E'.
- If the output is coded in EBCDIC, you cannot specify any of the following values for any delimiter: X'15', X'0D', X'25'.
- If the output is coded in EBCDIC DBCS or MBCS, you cannot specify any of the following values for character string delimiters: X'0D', X'15', X'25', X'4B'.

The following table lists by encoding scheme the default hex values for the delimiter characters.
Table 128. Default delimiter values for different encoding schemes

<table>
<thead>
<tr>
<th>Character</th>
<th>EBCDIC SBCS</th>
<th>EBCDIC DBCS/MBCS</th>
<th>ASCII/Unicode SBCS</th>
<th>ASCII/Unicode MBCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character string delimiter</td>
<td>X'7F'</td>
<td>X'7F'</td>
<td>X'22'</td>
<td>X'22'</td>
</tr>
<tr>
<td>Decimal point character</td>
<td>X'4B'</td>
<td>X'4B'</td>
<td>X'2E'</td>
<td>X'2E'</td>
</tr>
<tr>
<td>Column delimiter</td>
<td>X'6B'</td>
<td>X'6B'</td>
<td>X'2C'</td>
<td>X'2C'</td>
</tr>
</tbody>
</table>

In most EBCDIC code pages, the hex values in the previous table represent a double quotation mark ("), a period (.) for the decimal point character, and a comma (,) for the column delimiter.

The following table lists by encoding scheme the maximum allowable hex values for any delimiter character.

Table 129. Maximum delimiter values for different encoding schemes

<table>
<thead>
<tr>
<th>Encoding scheme</th>
<th>Maximum allowable value</th>
</tr>
</thead>
<tbody>
<tr>
<td>EBCDIC SBCS</td>
<td>None</td>
</tr>
<tr>
<td>EBCDIC DBCS/MBCS</td>
<td>X'3F' (X'7F', X'4B', X'6B' are also allowed.)</td>
</tr>
<tr>
<td>ASCII/Unicode SBCS</td>
<td>None</td>
</tr>
<tr>
<td>ASCII/Unicode MBCS</td>
<td>X'7F'</td>
</tr>
</tbody>
</table>

The following table identifies the acceptable data type forms for the delimited file format that the LOAD and UNLOAD utilities use.

Table 130. Acceptable data type forms for delimited files

<table>
<thead>
<tr>
<th>Data type</th>
<th>Acceptable form for loading a delimited file</th>
<th>Form that is created by unloading a delimited file</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR, VARCHAR</td>
<td>A delimited or non-delimited character string</td>
<td>Character data that is enclosed by character delimiters. For VARCHAR, length bytes do not precede the data in the string.</td>
</tr>
<tr>
<td>GRAPHIC (any type)</td>
<td>A delimited or non-delimited character stream</td>
<td>Data that is unloaded as a delimited character string. For VARGRAPHIC, length bytes do not precede the data in the string.</td>
</tr>
<tr>
<td>INTEGER (any type)</td>
<td>A stream of characters that represents a number in EXTERNAL format</td>
<td>Numeric data in external format.</td>
</tr>
<tr>
<td>Decimal (any type)</td>
<td>A character stream that represents a number in EXTERNAL format</td>
<td>A string of characters that represents a number.</td>
</tr>
<tr>
<td>FLOAT</td>
<td>Representation of a number in the range -7.2E + 75 to 7.2E + 75 in EXTERNAL format</td>
<td>A string of characters that represents a number in floating-point notation.</td>
</tr>
<tr>
<td>BLOB, CLOB</td>
<td>A delimited or non-delimited character string</td>
<td>Character data that is enclosed by character delimiters. Length bytes do not precede the data in the string.</td>
</tr>
</tbody>
</table>
Table 130. Acceptable data type forms for delimited files (continued)

<table>
<thead>
<tr>
<th>Data type</th>
<th>Acceptable form for loading a delimited file</th>
<th>Form that is created by unloading a delimited file</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBCLOB</td>
<td>A delimited or non-delimited character string</td>
<td>Character data that is enclosed by character delimiters. Length bytes do not precede the data in the string.</td>
</tr>
<tr>
<td>DATE</td>
<td>A delimited or non-delimited character string that contains a date value in EXTERNAL format</td>
<td>A string of characters that represents a date.</td>
</tr>
<tr>
<td>TIME</td>
<td>A delimited or non-delimited character string that contains a time value in EXTERNAL format</td>
<td>A string of characters that represents a time.</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>A delimited or non-delimited character string that contains a timestamp value in EXTERNAL format</td>
<td>A string of characters that represents a timestamp.</td>
</tr>
<tr>
<td>XML</td>
<td>A delimited or non-delimited XML character string</td>
<td>A string of characters that represents an XML document.</td>
</tr>
</tbody>
</table>

**Related concepts:**
- "Loading delimited files" on page 325

**Related reference:**
- Appendix G, “Delimited file format,” on page 1125

### Specifying TRUNCATE and STRIP options for output data

You can unload certain types of data into output fields that are shorter than the length of the output data. This data truncation occurs only when you explicitly specify the TRUNCATE option. Any CCSID conversion is applied first, and then truncation is applied to encoded data for output.

For bit strings, truncation occurs at a byte boundary. For character type data, truncation occurs at a character boundary (a multi-byte character is not split). If a mixed-character type data is truncated in an output field of fixed size, the truncated string can be shorter than the specified field size. In this case, blanks in the output CCSID are padded to the right. If the output data is in EBCDIC for a mixed-character type field, truncation preserves the SO (shift-out) and the SI (shift-in) characters around a DBCS substring.

The TRUNCATE option of the UNLOAD utility truncates string data, and it has a different purpose than the SQL TRUNCATE scalar function.

For VARCHAR and VARGRAPHIC, and VARBINARY output fields, in addition to the TRUNCATE option, the STRIP option is provided to remove the specified characters, or the leading blanks, the trailing blanks, or both. The strip operation is applied on the encoded data for output. If both the TRUNCATE and STRIP options are specified, the truncation operation is applied first, and then strip is applied. For example, the output for an UNLOAD job in which you specify both the TRUNCATE and STRIP options for a VARCHAR(5) output field is shown in the following table. In this table, an underscore represents a character that is to be stripped. In all cases, the source string is first truncated to ' _ABC_' (a five-character string to fit in the VARCHAR(5) field), and then the strip operation is applied.
Table 131. Results of specifying both the TRUNCATE and STRIP options for UNLOAD

<table>
<thead>
<tr>
<th>Specified STRIP option</th>
<th>Source string</th>
<th>Truncated string</th>
<th>Output string</th>
<th>Specified length</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRIP BOTH</td>
<td>'ABC_DEF'</td>
<td>'ABC_'</td>
<td>'ABC'</td>
<td>3</td>
</tr>
<tr>
<td>STRIP LEADING</td>
<td>'ABC_DEF'</td>
<td>'ABC_'</td>
<td>'ABC_'</td>
<td>4</td>
</tr>
<tr>
<td>STRIP TRAILING</td>
<td>'ABC_DEF'</td>
<td>'ABC_'</td>
<td>'ABC'</td>
<td>4</td>
</tr>
</tbody>
</table>

The following control statement shows an example of using the STRIP option.

In the example, STRIP TRAILING ' _ ' is included in the field specification for the TEXT column. The TEXT column contains variable character data with a maximum length of 8 characters, as specified by VARCHAR(8). When the UNLOAD utility unloads the table, all occurrences of the '_' (underscore) character at the end of the data from the TEXT column are stripped from the data.

```
UNLOAD TABLESPACE DB.TS
  PUNCHDDN SYSPUNCH
  UNLDDN UNLDD2
  FROM TABLE TB
    (EMPNO POSITION(*) CHAR(6),
    TEXT POSITION(*) VARCHAR(8) STRIP TRAILING ' _ ',
    DEPTNO POSITION(*) CLOB(4),
    ROWID POSITION(*) ROWID,
    LAST_UPDATE POSITION(*) TIME EXTERNAL)
```

The following table further illustrates the STRIP option.

Table 132. Example of the results of specifying the STRIP option for UNLOAD

<table>
<thead>
<tr>
<th>Original data</th>
<th>STRIP specification</th>
<th>Data after stripping</th>
<th>Final length</th>
</tr>
</thead>
<tbody>
<tr>
<td>'ABC_'</td>
<td>STRIP LEADING ' _ '</td>
<td>'ABC_'</td>
<td>4</td>
</tr>
<tr>
<td>'ABC_'</td>
<td>STRIP TRAILING ' _ '</td>
<td>' _ABC'</td>
<td>4</td>
</tr>
<tr>
<td>' <em>ABC</em>'</td>
<td>STRIP BOTH ' _ '</td>
<td>'ABC'</td>
<td>3</td>
</tr>
</tbody>
</table>

LOAD statements that are generated by UNLOAD

The UNLOAD utility can generate one or more LOAD utility statements that you can later use to load the unloaded data into either the original table or different tables. The generated LOAD statements are written to the data set that is specified by the PUNCHDDN option. By default, that data set is SYSPUNCH.

To request that UNLOAD generate a LOAD statement, perform one of the following actions:

- Include a SYSPUNCH DD statement or SYSPUNCH template in the utility job.
- Specify the PUNCHDDN option with a different DD name or template name and include that DD statement or template in the utility job.

If PUNCHDDN is not specified and the SYSPUNCH DD name does not exist, the LOAD statement is not generated.

If multiple table spaces are to be unloaded and you want UNLOAD to generate a LOAD statement for each table space, specify a physically distinct data set for each table space. To do so, specify the PUNCHDDN option with a template that contains the table space as a variable (&TS).
Make any necessary changes to the generated LOAD statement. For example, by default, the generated LOAD statement includes WHEN and INTO TABLE specifications that identify the table where the rows are to be reloaded. These specifications are not included if the HEADER NONE option was specified in the UNLOAD statement. If you intend to load the UNLOAD output data into different tables than the original ones, you need to edit the generated LOAD statement.

Related tasks:

“Loading tables with special column types by using generated LOAD statements” on page 318

Related reference:

Chapter 34, “TEMPLATE,” on page 793

Unloading compressed data

You can unload compressed rows from an image copy data set only when the dictionary for decompression has been retrieved. If a row is compressed and the dictionary pages have not been read when the row is encountered, the UNLOAD utility ignores this row, issues a warning message, and increments the error count.

About this task

If the error count reaches the limit that is specified by the MAXERR option, UNLOAD terminates with an error message.

If the image copy data set is an incremental copy or a copy of pieces that does not contain a dictionary, the FROMCOPYDDN option can be used for a DD name to concatenate the data set with the corresponding full image copy that contains the dictionary. If SYSTEMPAGES YES is used, a dictionary will always be available in the incremental copies or pieces.

Field specification errors

If the UNLOAD utility detects any inconsistency relating to the field specification, Db2 issues an error message. For example, the UNLOAD utility might detect a data conversion problem or an encoding problem that occurs during the unloading of a row.

If the MAXERR option specifies a number that is greater than zero, the UNLOAD utility continues processing until the total number of the records in error reaches the specified MAXERR number. Db2 issues one message for each record in error and does not unload the record.

Sample UNLOAD control statements

Use the sample control statements as models for developing your own UNLOAD control statements.

Example 1: Unloading all columns of specified rows

The control statement specifies that all columns of rows that meet the following criteria are to be unloaded from table DSN8810.EMP in table space DSN8D11A.DSN8S71E:

- The value in the WORKDEPT column is D11.
- The value in the SALARY column is greater than 25,000.
Example 2: Unloading specific columns by using a field specification list

The following control statement specifies that columns EMPNO, LASTNAME, and SALARY are to be unloaded, in that order, for all rows that meet the specified conditions. These conditions are specified in the WHEN clause and are the same as those conditions in example 1. The SALARY column is to be unloaded as type DECIMAL EXTERNAL. The NOPAD option indicates that variable-length fields are to be unloaded without any padding.

```sql
UNLOAD TABLESPACE DSN8D11A.DSN8S81E NOPAD FROM TABLE DSN8B10.EMP WHEN (WORKDEPT = 'D11' AND SALARY > 25000)
```

The output from this example might look similar to the following output:

```
000060@@STERN# 32250.00
000150@@ADAMSON# 25280.00
000200@@BROWN# 27740.00
000220@@LUTZ# 29840.00
200220@@JOHN# 29840.00
```

In this output:

- `@@` before the last name represents the 2-byte binary field that contains the length of the VARCHAR field LASTNAME (for example, X'0005' for STERN).
- `#` represents the NULL indicator byte for the nullable SALARY field.
- Because the SALARY column is declared as DECIMAL (9,2) on the table, the default output length of the SALARY field is 11 (9 digits + sign + decimal point), not including the NULL indicator byte.
- LASTNAME is unloaded as a variable-length field because the NOPAD option is specified.

Example 3: Unloading data from an image copy

The FROMCOPY option in the following control statement specifies that data is to be unloaded from a single image copy data set, JUKWU111.FCOPY1.STEP1.FCOPY1.

```sql
PUNCHDDN SYSPUNCH specifies that the UNLOAD utility is to generate LOAD utility control statements and write them to the data set that is defined by the SYSPUNCH DD statement; SYSPUNCH is the default. UNLDDN SYSREC specifies that the data is to be unloaded to the data set that is defined by the SYSREC DD statement; SYSREC is the default.
```
Example 4: Unloading a sample of rows and specifying a header.

The following control statement specifies that a sample of rows is to be unloaded from table ADMF001.TBKW1605. Unloading a sample of rows is useful for building a test system. The SAMPLE option indicates that 75% of the rows are to be sampled. The HEADER option indicates that the string 'sample' is to be used as the header field in the output file. The PUNCHDDN option indicates that UNLOAD is to generate LOAD utility control statements and write them to the SYSPUNCH data set, which is the default. UNLOAD specifies the header field as a criterion in the WHEN clause of these LOAD statements.

Example 5: Unloading data from two tables in a segmented table space

The following control statement specifies that data from table ADMF001.TBKW1504 and table ADMF001.TBKW1505 is to be unloaded from the segmented table space DBKW1502.TSKW1502. The PUNCHDDN option indicates that UNLOAD is to generate LOAD utility control statements and write them to the SYSPUNCH data set, which is the default. The UNLDDN option specifies that the data is to be unloaded to the data set that is defined by the SYSREC DD statement, which is also the default.

Example 6: Unloading data in parallel from a partitioned table space

The UNLOAD control statement specifies that data from table TCRT.TTBL is to be unloaded to data sets that are defined by the UNLDDS template. These data sets are to be dynamically allocated and named according to the naming convention that is defined by the DSN option of the TEMPLATE utility control statement. This naming convention indicates that a data set is to be allocated for each table space partition.

Assume that table space TDB1.TSP1, which contains table TCRT.TTBL, has three partitions. Because the table space is partitioned and each partition is associated with an output data set that is defined by the UNLDDS template, the UNLOAD job runs in parallel in a multi-processor environment. The number of parallel tasks are determined by the number of available processors.
Example 7: Using a LISTDEF utility statement to specify partitions to unload

The UNLOAD control statement specifies that data that is included in the UNLDLIST list is to be unloaded. UNLDLIST is defined in the LISTDEF utility control statement and contains partitions one and three of table space TDB1.TSP1. The LIST option of the UNLOAD statement specifies that the UNLOAD utility is to use this list.

The data is to be unloaded to data sets that are defined by the UNLDDS template.

Example 8: Unloading multiple table spaces by using LISTDEF

The UNLOAD control statement specifies that data from multiple table spaces is to be unloaded. These table spaces are specified in the LISTDEF utility control statement. Assume that the database TDB1 contains two table spaces that can be expressed by the pattern-matching string 'TSP*', (for example, TSP1 and TSP2). These table spaces are both included in the list named UNLDLIST, which is
defined in the LISTDEF statement. The LIST option of the UNLOAD statement specifies that the UNLOAD utility is to use this list.

The UNLDDN option specifies that the data is to be unloaded to data sets that are defined by the UNLDDS template. The PUNCHDDN option specifies that UNLOAD is to generate LOAD utility control statements and write them to the data sets that are defined by the PUNCHDS template.

```
//SAMPJOB JOB ...
//STEP1 EXEC DSNUPROC,UID='SMPLUNLD',UTPROC='',SYSTEM='DSN'
//SYSPRINT DD SYSOUT=* 
//SYSIN DD *
   LISTDEF UNLDLIST
   INCLUDE TABLESPACE TDB1.TSP*
   TEMPLATE UNLDDS DSN &USERID..SMPLUNLD.&TS.
   UNIT SYSDA DISP (NEW,CATLG,CATLG) SPACE (2,1) CYL
   TEMPLATE PUNCHDS DSN &USERID..SMPLPUNC.&TS.
   UNIT SYSDA DISP (NEW,CATLG,CATLG) SPACE (1,1) CYL
   UNLOAD LIST UNLDLIST
   PUNCHDDN PUNCHDS -- TEMPLATE name
   UNLDDN UNLDDS -- TEMPLATE name
```

Figure 89. Example of unloading multiple table spaces

Assume that the user ID is USERID. This UNLOAD job creates the following two data sets to store the unloaded data:
- USERID.SMPLUNLD.TSP1 ... contains rows from table space TDB1.TSP1.
- USERID.SMPLUNLD.TSP2 ... contains rows from table space TDB1.TSP2.

Example 9: Unloading data into a delimited file.

The control statement specifies that data from the specified columns (RECID, CHAR7SBCS, CHAR7BIT, VCHAR20, VCHAR20SBCS, VCHAR20BIT) in table TBQB0501 is to be unloaded into a delimited file. This output format is indicated by the DELIMITED option. The POSITION(*) option indicates that each field in the output file is to start at the first byte after the last position of the previous field.

The column delimiter is specified by the COLDEL option as a semicolon (;), the character string delimiter is specified by the CHARDEL option as a pound sign (#), and the decimal point character is specified by the DECPT option as an exclamation point (!).

PUNCHDDN SYSPUNCH specifies that UNLOAD is to generate LOAD utility control statements and store them in the SYSPUNCH data set, which is the default. UNLDDN SYSREC indicates that the data is to be unloaded to the SYSREC data set, which is the default.

The EBCDIC option indicates that all output character data is to be in EBCDIC.
Example 10: Converting character data

For this example, assume that table DSN8810.DEMO_UNICODE contains character data in Unicode. The UNLOAD control statement specifies that the utility is to unload the data in this table as EBCDIC data.

Example 11: Unloading LOB data to a file

The UNLOAD control statement specifies that the utility is to unload data from table DSN8910.EMP_PHOTO_RESUME into the data set that is identified by the SYSREC DD statement. Data in the EMPNO field is six bytes of character data, as indicated by the CHAR(6) option, and is unloaded directly into the SYSREC data set. Data in the RESUME column is CLOB data as indicated by the CLOBF option. This CLOB data is to be unloaded to the files identified by the LOBFRV template, which is defined in the preceding TEMPLATE statement. If these files do not already exist, Db2 creates them. The names of these files are stored in the SYSREC data set. The length of the file name to be stored in this data set can be up to 255 bytes as specified by the VARCHAR option.

Figure 90. Example of unloading data into a delimited file.

Example 91. Example of unloading Unicode table data into EBCDIC

Figure 92. Example of unloading LOB data into a file
Example 12: Unloading data from clone tables

The UNLOAD control statement specifies that the utility is to unload data from only clone tables in the specified table spaces. The PUNCHDDN option specifies that the SYSPUNCH data set is to receive the LOAD utility control statements that the UNLOAD utility generates.

```
UNLOAD TABLESPACE DBKQRE01.TPKQRE01
    FROM TABLE ADMFO01.TBKQRE01.Clone
    PUNCHDDN SYSPUNCH UNLDDN SYSP
    CLONE
```
Part 3. Db2 stand-alone utilities

The stand-alone utilities run as batch jobs that are independent of Db2. The only way to run these utilities is to use JCL.
Chapter 36. Invoking stand-alone utilities

To invoke a stand-alone utility, you must use JCL. Some stand-alone utilities read
the utility control statements from an input stream; other utilities obtain the
function definitions from JCL EXEC PARM parameters.

Before you begin

Make sure that you are authorized to run the utility that you want to invoke. To
check the required authorization, see the following information for the utility that
you want to invoke.

Procedure

To invoke a stand-alone utility:
1. Prepare the necessary data sets. For information about the data sets that are
required for the utility that you want to invoke, see the information for that
utility:

   - Chapter 37, “DSNJCNVB,” on page 895
   - Chapter 38, “DSNJCNVT,” on page 897
   - Chapter 39, “DSNJLOGF (preformat active log),” on page 901
   - Chapter 40, “DSNJU003 (change log inventory),” on page 903
   - Chapter 41, “DSNJU004 (print log map),” on page 927
   - Chapter 43, “DSN1COMP,” on page 949
   - “Data sets that DSN1COPY uses” on page 972
   - Chapter 45, “DSN1LOGP,” on page 989
   - Chapter 46, “DSN1PRNT,” on page 1009
   - Chapter 47, “DSN1SDMP,” on page 1021

2. Invoke the utility by creating a JCL job with the following information:

   - Specify the utility name on the EXEC statement, in the PGM parameter.
   - Specify the utility options as follows:

     **DSN1COMP, DSN1COPY, and DSN1PRNT**
     For these utilities, specify the utility options on the EXEC statement,
in the PARM parameter. For more information about how to specify
these options and sample JCL for each utility, see “Specifying options
for stand-alone utilities by using the JCL EXEC PARM parameter” on
page 892.

     **DSNJU003, DSNJU004, DSN1LOGP, and DSN1SDMP**
     For these utilities, create a utility control statement in an in-stream
data set. For more information about these utility control statements,
the DD name to use for the in-stream data set, and sample JCL for
each utility, see “Stand-alone utility control statements” on page 892.

     **DSNJCNVB, DSNJCNVT, and DSNJLOGF**
     These utilities do not have any options. For sample JCL for these
utilities, see the following information:

        - Chapter 37, “DSNJCNVB,” on page 895
        - Chapter 38, “DSNJCNVT,” on page 897
Specifying options for stand-alone utilities by using the JCL EXEC PARM parameter

Use the EXEC PARM parameter to specify function options for the following stand-alone utilities: DSN1COMP, DSN1COPY, and DSN1PRNT.

Procedure

To specify options for stand-alone utilities by using the JCL EXEC PARM parameter:

• Ensure that the parameters that you specify obey the following JCL EXEC PARM parameter specification rules:
  – Enclose multiple subparameters in single quotation marks or parentheses and separate the subparameters with commas, as in the following example:
    //name EXEC PARM='ABC,...,XYZ'
  – Ensure that the total length of the parameter specification does not exceed 100 characters.
  – Do not use blanks within the parameter specification.

• To specify the parameter across multiple lines:
  1. Enclose it in parentheses.
  2. End the first line with a subparameter, followed by a comma.
  3. Continue the subparameters on the next line, beginning before column 17.

The following example shows a parameter that spans multiple lines:
//stepname EXEC PARM=(ABC,...LMN,
OPQ,...,XYZ)

Example

For sample JCL, see the following information:

• “Sample DSN1COMP control statements” on page 956
• “Sample DSN1COPY control statements” on page 983
• “Sample DSN1PRNT control statements” on page 1017

Related reference:

EXEC PARM Parameter (MVS JCL Reference)

Stand-alone utility control statements

Utility control statements include parameters that define the function that a utility job performs. Only some stand-alone utilities require utility control statements. For the other stand-alone utilities, you specify the utility parameters in the PARM parameter of the EXEC statement.
The following table lists the stand-alone utilities that read utility control statements from an in-stream data set. Use the listed DD name for the in-stream data set.

Table 133. Stand-alone utilities that read utility control statements

<table>
<thead>
<tr>
<th>Utility</th>
<th>DD name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNJU003 (change log inventory)</td>
<td>SYSIN</td>
</tr>
<tr>
<td>DSNJU004 (print log map)</td>
<td>SYSIN (optional)</td>
</tr>
<tr>
<td>DSN1LOGP</td>
<td>SYSIN</td>
</tr>
<tr>
<td>DSN1SDMP</td>
<td>SDMPIN</td>
</tr>
</tbody>
</table>

The statements must conform to the following rules:
- The logical record length (LRECL) must be 80 characters. Columns 73 through 80 are ignored.
- The records are concatenated into a single stream before they are parsed. No concatenation character is necessary.
- The SYSIN stream can contain multiple utility control statements.

For sample JCL with utility control statements, see the following information:
- “Sample DSNJU003 control statements” on page 924
- “Sample DSNJU004 control statement” on page 929
- “Sample DSN1LOGP control statements” on page 999
- “Sample DSN1SDMP control statements” on page 1029

Related concepts:
- JCL DD statement: ddnames that are reserved for specific uses (z/OS basic skills)
- Coding JCL: Data set types and name syntax (z/OS Basic Skills)

Related tasks:
- “Specifying options for stand-alone utilities by using the JCL EXEC PARM parameter” on page 892
Chapter 37. DSNJCNVB

The DSNJCNVB stand-alone conversion utility converts the bootstrap data set (BSDS) so that it can support up to 10,000 archive log volumes and 93 active log data sets per log copy.

Environment

Execute the DSNJCNVB utility as a batch job only when Db2 is not running.

Authorization required

The authorization ID of the DSNJCNVB job must have the requisite RACF authorization.

Prerequisite actions

If you have migrated to a new version of Db2, you need to create a larger BSDS before converting it. For a new installation, you do not need to create a larger BSDS. Db2 provides a larger BSDS definition in installation job DSNTIJIN; however, if you want to convert the BSDS, you must still run DSNJCNVB.

Required and optional data sets

DSNJCNVB recognizes DD statements with the following DD names:

SYSUT1

Specifies the BSDS copy 1 data set that DSNJCNVB is to use as input. This statement is required.

SYSUT2

Specifies the BSDS copy 2 data set that DSNJCNVB is to use as input. This statement is optional.

Specify this statement if you are using dual BSDSs and you want to convert both with a single execution of DSNJCNVB. You can run DSNJCNVB separately for each copy.

SYSPRINT

Specifies a data set or print spool class for print output. This statement is required. The logical record length (LRECL) is 125.

Running DSNJCNVB

Use the following EXEC statement to execute this utility:

//EXEC PGM=DSNJCNVB

Sample DSNJCNVB control statement

The following statements specify that DSNJCNVB is to convert the BSDS so that it can manage up to 10,000 archive log volumes and 93 active log data sets per log copy. The SYSUT1 and SYSUT2 statements identify the bootstrap data sets. Only the SYSUT1 statement is required. The SYSUT2 statement is optional. Specify SYSUT2 only if you are using dual BSDSs and you want to convert both with a single execution of DSNJCNVB.
DSNJCNVB output

The following example shows sample DSNJCNVB output:

CONVERSION OF BSDS DATA SET - COPY 1, DSN=DSNC810.BSDS01
SYSTEM TIMESTAMP - DATE=2003.199  LTIME= 9:40:58.74
UTILITY TIMESTAMP - DATE=2003.216  LTIME=14:26:02.21
PREVIOUS HIKEY - 04000053
NEW HIKEY - 040002F0
RECORDS ADDED - 669

DSNJ260I  DSNJCNVB BSDS CONVERSION FOR DDNAME=SYSUT1 COMPLETED SUCCESSFULLY
DSNJ200I  DSNJCNVB CONVERT BSDS UTILITY PROCESSING COMPLETED SUCCESSFULLY

Related tasks:

Adding a second BSDS (Db2 Installation and Migration)
Chapter 38. DSNJCNVT

The DSNJCNVT stand-alone conversion utility converts the bootstrap data set (BSDS) records that are necessary to support 10-byte RBA and LRSN fields.

At migration to Db2 12, you cannot start Db2 12 until the BSDS is converted to use the 10-byte RBA and LRSN formats. You can convert the BSDS before or during the Db2 12 migration process.

Running DSNJCNVT is optional and can be done any time after migrating to Db2 11 new-function mode. Conversion is required during installation if the RBA or LRSN is approaching the end of the range. For a data-sharing installation, if the LRSN is approaching the end of the range, the BSDS of each member can be converted one at a time.

In data sharing, you must convert the BSDS for each Db2 member separately.

Attention: In Db2 subsystems that are not data sharing members, if Db2 is already at risk of reaching the 6-byte RBA limit, it is strongly recommended that you first convert all catalog and directory objects, then convert all user objects to the 10-byte RBA format, before you convert the BSDS.

In Db2 subsystems that are not data sharing members, always convert all Db2 catalog, directory, and user objects to use the extended 10-byte RBA format before you convert the BSDS, especially if Db2 is close to reaching the logging limit for the 6-byte RBA. Failure to convert page sets to the 10-byte RBA format before Db2 reaches the 6-byte logging limit results in failed updates with reason code 00C2026D. No updates are allowed for any object that is still in the 6-byte format.

In data sharing, if any Db2 member is approaching the logging limit for the 6-byte RBA but the LRSN is not approaching the limit of the 6-byte range, converting the BSDS of just that member sufficient to resolve the immediate problem and prevent outages. However, if the LRSN is also approaching the end of the 6-byte range, you must continue and convert page sets to use the 10-byte format before the limit is reached.

For instructions, see Converting the BSDS to the 10-byte RBA and LRSN (Db2 Administration Guide).

Environment

Run the DSNJCNVT utility as a batch job only when Db2 is not running.

Your Db2 subsystem must be in new-function mode to convert the BSDS.

Authorization required

The authorization ID of the DSNJCNVT job must have the required RACF authorization and read/write access to the new BSDSs and read access to the old BSDSs.
Required and optional data sets

DSNJCNVT recognizes DD statements with the following DD names:

SYSUT1
   Specifies the old BSDS that is to be converted. This statement is required.

SYSUT2
   Specifies the second copy of the old BSDS that is to be converted. This statement is optional.

SYSUT3
   Specifies the new, converted BSDS. This statement is required.

SYSUT4
   Specifies the second copy of the converted BSDS. This statement is required if the installation uses dual BSDSs; otherwise, it is optional.

SYSPRINT
   Contains the output messages from the conversion utility. This statement is required.

Running DSNJCNVT

Use the following EXEC statement to run this utility:

//EXEC   PGM=DSNJCNVT

Considerations for running DSNJCNVT

- The Db2 subsystem that owns the BSDSs that are to be converted must be stopped. DSNJCNVT is a stand-alone utility.
- In a data-sharing environment, allow Db2 utilities that read the logs of peer members to finish before converting the BSDSs.
- In a data-sharing environment, stop data replication products before the conversion to ensure that the old BSDSs can be successfully renamed and replaced by the converted BSDSs. The recommended procedure is to stop the replication product first and then stop the Db2 system that is to have its BSDSs converted. This procedure allows sharing systems to deallocate the BSDSs when the state of the member changes to inactive.
- The RACF user ID that is running DSNJCNVT must have read/write access to the new BSDSs and read access to the old BSDSs.
- The Db2 subsystem that owns the BSDS that is to be converted must start after the data sharing group was migrated to Db2 11 new-function mode.
- Conversion to the new BSDS format is required to write new format log records and remove the 6-byte RBA and LRSN limits.

Sample DSNJCNVT control statement

The following statements specify that DSNJCNVT is to convert the BSDS that is needed to support 10-byte RBA and LRSN fields.

//CONVERT  EXEC  PGM=DSNJCNVT,REGION=64M
//SYSUT1   DD  DSN=DB2A.OLD.BSDS01,DISP=SHR
//SYSUT2   DD  DSN=DB2A.OLD.BSDS02,DISP=SHR
//SYSUT3   DD  DSN=DB2A.BSDS01,DISP=OLD
//SYSUT4   DD  DSN=DB2A.BSDS02,DISP=OLD
//SYSPRINT DD  SYSOUT=*
You can view another example of using DSNJCNVT in job DSNTIJCB in
prefix.SDSSAMP.

**DSNJCNVT output**

The following example shows sample DSNJCNVT output:

```plaintext
CRCR convert started
DSNJ200I DSNJCNVT CONVERT UTILITY PROCESSING COMPLETED SUCCESSFULLY
FOR MEMBER 'xxxxxxxxx'
```

**Related concepts:**

- How RBA and LRSN values are displayed (Db2 Administration Guide)
- The extended 10-byte RBA and LRSN in Db2 11 (Db2 for z/OS What's New?)

**Related tasks:**

- What to do before RBA or LRSN limits are reached (Db2 Administration Guide)
Chapter 39. DSNJLOGF (preformat active log)

When writing to an active log data set for the first time, Db2 must preformat a VSAM control area before writing the log records. The DSNJLOGF stand-alone utility avoids this delay by preformatting the active log data sets before bringing them online to Db2.

The following EXEC statement is used to invoke DSNJLOGF:

```
//stepname EXEC PGM=DSNJLOGF
```

**Environment**

Run DSNJLOGF as a z/OS job.

**Required and optional data sets**

All SYSUTx DD statements are optional, but at least one must be specified.

DSNJLOGF recognizes DD statements with the following DD names.

- **SYSUT0**
  - Defines the newly defined active log data set that is to be preformatted. The data set must be an empty VSAM linear data set and less than four gigabytes in size.

- **SYSUT1**
  - Defines a newly defined active log data set that is to be preformatted. This statement is optional.

- **SYSUT2**
  - Defines a newly defined active log data set that is to be preformatted. This statement is optional.

- **SYSUT3**
  - Defines a newly defined active log data set that is to be preformatted. This statement is optional.

- **SYSUT4**
  - Defines a newly defined active log data set that is to be preformatted. This statement is optional.

- **SYSUT5**
  - Defines a newly defined active log data set that is to be preformatted. This statement is optional.

- **SYSUT6**
  - Defines a newly defined active log data set that is to be preformatted. This statement is optional.

- **SYSUT7**
  - Defines a newly defined active log data set that is to be preformatted. This statement is optional.

- **SYSUT8**
  - Defines a newly defined active log data set that is to be preformatted. This statement is optional.
SYSUT9
Defines a newly defined active log data set that is to be preformatted. This statement is optional.

SYSPRINT
Defines the print spool class or data set for print output. The logical record length (LRECL) is 132.

Sample DSNJLOGF control statement

The control statements in the following figure specify that DSNJLOGF is to preformat the active log data sets that are identified by the DD statements.

```plaintext
//MULTFMT EXEC PGM=DSNJLOGF,REGION=64M
//SYSPRINT DD SYSOUT=* 
//SYSUT0 DD DSN=DSNTEMP.LOGCOPY1.DS00,DISP=SHR
//SYSUT1 DD DSN=DSNTEMP.LOGCOPY1.DS01,DISP=SHR
//SYSUT2 DD DSN=DSNTEMP.LOGCOPY1.DS02,DISP=SHR
//SYSUT3 DD DSN=DSNTEMP.LOGCOPY1.DS03,DISP=SHR
//SYSUT4 DD DSN=DSNTEMP.LOGCOPY1.DS04,DISP=SHR
//SYSUT5 DD DSN=DSNTEMP.LOGCOPY1.DS05,DISP=SHR
//SYSUT6 DD DSN=DSNTEMP.LOGCOPY1.DS06,DISP=SHR
//SYSUT7 DD DSN=DSNTEMP.LOGCOPY1.DS07,DISP=SHR
//SYSUT8 DD DSN=DSNTEMP.LOGCOPY1.DS08,DISP=SHR
//SYSUT9 DD DSN=DSNTEMP.LOGCOPY1.DS09,DISP=SHR
```

Figure 93. Sample DSNJLOGF control statement

DSNJLOGF output

The following sample shows the DSNJLOGF output for the second data set in the previous sample control statement shown above.
```
DSNJ991I DSNJLOGF START OF LOG DATASET PREFORMAT FOR JOB LOGFRMT STEP1
DSNJ992I DSNJLOGF LOG DATA SET NAME = DSNC110.LOGCOPY1.DS01
DSNJ996I DSNJLOGF LOG PREFORMAT COMPLETED SUCCESSFULLY, 00015000 RECORDS FORMATTED
```
Chapter 40. DSNJU003 (change log inventory)

The DSNJU003 stand-alone utility changes the bootstrap data sets (BSDSs).

You can use the utility to:

- Add or delete active or archive log data sets
- Add or delete checkpoint records
- Create a conditional restart control record to control the next start of the Db2 subsystem
- Change the VSAM catalog name entry in the BSDS
- Modify the communication record in the BSDS
- Modify the value for the highest-written log RBA value (relative byte address within the log) or the highest-offloaded RBA value
- Deactivate a member of a data sharing group
- Destroy a member from a data sharing group
- Reactivate a deactivated member of a data sharing group

Environment

Execute the change log inventory utility only as a batch job when Db2 is not running. Changing a BSDS for a data-sharing member by using DSNJU003 might cause a log read request from another data-sharing member to fail. The failure occurs only if the second member tries to access the changed BSDS before the first member is started.

Authorization required

The authorization ID of the DSNJU003 job must have the requisite RACF authorization.

Required and optional data sets

DSNJU003 recognizes DD statements with the following DD names:

**SYSUT1**

Specifies and allocates the bootstrap data set. This statement is required.

**SYSUT2**

Specifies and allocates a second copy of the bootstrap data set. This statement is required if you use dual BSDSs.

**Dual BSDSs and DSNJU003**: With each execution of DSNJU003, the BSDS timestamp field is updated with the current system time. If you run DSNJU003 separately for each copy of a dual copy BSDS, the timestamp fields are not synchronized, and Db2 fails at startup. If you change the contents of the BSDS copy by running DSNJU003, Db2 issues error message DSNJ122I. Therefore, if you use DSNJU003 to update dual copy BSDSs, update both BSDSs within a single execution of DSNJU003.

**SYSPRINT**

Specifies a data set for print output. This statement is required. The logical record length (LRECL) is 125.
SYSIN
    Specifies the input data set for statements. This statement is required. The logical record length (LRECL) is 80.

Running DSNJU003

Execute the utility with the following statement, which can be included only in a batch job:
//EXEC   PGM=DSNJU003

DSNJU003 utility control statements

DSNJU003 uses multiple statements that you submit in separate jobs. The statements are:
  • NEWLOG
  • DELETE
  • CRESTART
  • NEWCAT
  • DDF
  • CHECKPT
  • HIRESHBA
  • DELMBR
  • RSTMBR

DSNJU003 (change log inventory) syntax diagram

NEWLOG statement

new active log:

new archive log:
ip-spec:

(1)
- LOCATION=locname
- PORT=port
- RESPONSE=resport
- SEC_PORT=secport
- ALIAS=alias-name
- IPNAME=ipname

(2)
- IPV4=IPV4-address
- IPV6=IPV6-address
- GRP_IPV4=group-ipv4-addr
- GRP_IPV6=group-ipv6-addr

Notes:
1. You can specify each option (such as LOCATION or PORT) only once.
2. You can specify each option (IPV4 and IPV6) only once.

lu-spec:

(1)
- LOCATION=locname
- LUNAME=luname
- PASSWORD=password
- GENERIC=gluname
- PORT=port
- RESPONSE=resport
- ALIAS=alias-name
- :alias-port

Notes:
1. You can specify each option (such as LOCATION or PORT) only once.

no-spec:
CHECKPT statement

```
CHECKPT STARTRBA=startrba, ENDRBA=endrba, TIME=time, CANCEL, ENDLRSN=endlrsn
```

HIGHRBA statement

```
HIGHRBA STARTRBA=startrba, OFFLRBA=offlrba, TIME=time, OFFLRBA=offlrba
```

DELMBR statement

```
DELMBR DEACTIV, MEMBERID=member-id
```

RSTMBR statement

```
RSTMBR MEMBERID=member-id
```

DSNJU0003 option descriptions

**NEWLOG**

Declares one of the following data sets:

- A VSAM data set that is available for use as an active log data set.
  Use only the keywords DSNNAME=, COPY1, and COPY2.
- An active log data set that is replacing one that encountered an I/O error.
Use only the keywords DSNAME=, COPY1, COPY2, STARTRBA=, and ENDRBA=.

- An archive log data set volume.
  Use only the keywords DSNAME=, COPY1VOL=, COPY2VOL=, STARTRBA=, ENDRBA=, UNIT=, CATALOG=, STRTLRSN=, and ENDLRSN=.

If you create an archive log data set and add it to the BSDS with this utility, you can specify a name that Db2 might also generate. Db2 generates archive log data set names of the form DSNCA$ARCHLOGx.$A000000n where:
- DSNCA and ARCHLOG are parts of the data set prefix that you specified on installation panels DSNTIPA2 and DSNTIPH.
- x is 1 for the first copy of the logs, and 2 is for the second copy.
- A000000n represents the series of low-level qualifiers that Db2 generates for archive log data set names, beginning with A0000001, and incrementing to A0000002, A0000003, and so forth.

For data sharing, the naming convention is DSNCA$ARCHLOG1 or DSNCA$DSN1.ARCLG1.

If you do specify a name by using the same naming convention as Db2, you receive a dynamic allocation error when Db2 generates that name. The error message, DSNJ103I, is issued once. Db2 then increments the low-level qualifier to generate the next data set name in the series and offloads to it the next time Db2 archives. (The active log that previously was not offloaded is offloaded to this data set.)

The newly defined active logs cannot specify a start and end LRSN. When Db2 starts, it reads the new active log data sets with an RBA range to determine the LRSN range, and updates the start and end LRSN in the BSDS for the new log data sets. The start and end LRSN for new active logs that contain active log data are read at Db2 start-up time from the new active log data sets that are specified in the change log inventory NEWLOG statements. For new archive logs that are defined with change log inventory, the user must specify the start and end RBAs. For data sharing, the user must also specify the start and end LRSNs. Db2 startup does not attempt to find these values from the new archive log data sets.

**DSNAME= data-set-name**
- Specifies a log data set.
  - data-set-name can be up to 44 characters long.

**COPY1**
- Makes the data set an active log copy-1 data set.

**COPY2**
- Makes the data set an active log copy-2 data set.

**STARTRBA= startrba**
- Identifies a hexadecimal number of up to 20 characters. If you use fewer than 20 characters, leading zeros are added. startrba must end with '000'; otherwise Db2 returns a DSNJ4381 error message. You can obtain the RBA from messages or by printing the log map.

**On the NEWLOG statement**, startrba gives the log RBA of the beginning of the replacement active log data set or the archive log data set volume that is specified by DSNAME.
On the CRESTART statement, startrba is the earliest RBA of the log that is to be used during restart. If you omit STARTRBA, Db2 determines the beginning of the log range.

On the CHECKPT statement, startrba indicates the start checkpoint log record. STARTRBA is required when STARTTIME is specified.

On the HIGHRBA statement, startrba denotes the log RBA of the highest-written log record in the active log data sets.

**ENDRBA= endrba**

endrba is a hexadecimal number of up to 20 characters. If you use fewer than 20 characters, leading zeros are added. endrba must end with '000' or Db2 returns a DSNJ4381 error message.

On the NEWLOG statement, endrba gives the log RBA (relative byte address within the log) of the end of the replacement active log data set or the archive log data set volume that is specified by DSNAME.

On the CRESTART statement, endrba is the last RBA of the log that is to be used during restart, and it is also the starting RBA of the next active log that is written after restart. Any log information in the bootstrap data set, the active logs, and the archive logs with an RBA that is greater than endrba is discarded. If you omit ENDRBA, Db2 determines the end of the log range.

The value of ENDRBA must be a multiple of 4096. (The hexadecimal value must end in 000.) Also, the value must be greater than or equal to the value of STARTRBA. If STARTRBA and ENDRBA are equal, the next restart is a cold start; that is, no log records are processed during restart. The specified RBA becomes the beginning RBA of the new log.

On the CHECKPT statement, endrba indicates the end checkpoint log record that corresponds to the start checkpoint log record.

**COPY1VOL= vol-id**

vol-id is the volume serial of the copy-1 archive log data set that is specified after DSNAME.

**COPY2VOL= vol-id**

vol-id is the volume serial of the copy-2 archive log data set that is specified after DSNAME.

**UNIT= unit-id**

unit-id is the device type of the archive log data set that is named after DSNAME.

**CATALOG**

Indicates whether the archive log data set is to be cataloged.

**NO** Indicates that the archive log data set is not to be cataloged. All subsequent allocations of the data set are made using the unit and volume information that is specified on the statement.

**YES** Indicates that the archive log data set is to be cataloged. All subsequent allocations of the data set are made using the catalog.

Db2 requires that all archive log data sets on disk be cataloged. Select CATALOG=YES if the archive log data set is on disk.

**STRTLRSN= startlrsn**

On the NEWLOG statement, startlrsn identifies the LRSN in the log record.
header of the first complete log record on the new archive data set. startlrsn is a hexadecimal number of up to 20 characters. If you use fewer than 20 characters, leading zeros are added. In a data sharing environment, run the print log map utility to find an archive log data set and start and end RBAs and LRSNs.

ENDLRSN=endlrsn

endlrsn is a hexadecimal number of up to 20 characters. If you use fewer than 20 characters, leading zeros are added. In a data sharing environment, run the print log map utility to find an archive log data set and start and end RBAs and LRSNs.

For the NEWLOG and CHECKPT statements, the ENDLRSN option is valid only in a data sharing environment. For the CRESTART statement, the ENDLRSN option is valid in both data sharing and non-data sharing environments. This option cannot be specified with STARTRBA or ENDRBA.

On the NEWLOG statement, endlrsn is the LRSN in the log record header of the last log record on the new archive data set.

On the CRESTART statement, in a data sharing environment, endlrsn is an LRSN value that is to be used as the log truncation point. A valid log truncation point is any LRSN value for which there exists a log record with an LRSN that is greater than or equal to the specified LRSN value. Any log information in the bootstrap data set, the active logs, and the archive logs with an LRSN greater than endlrsn is discarded. If you omit ENDLRSN, Db2 determines the end of the log range.

In a non-data sharing environment, endlrsn is the RBA value that matches the start of the last log record that is to be used during restart. Any log information in the bootstrap data set, the active logs, and the archive logs with an RBA that is greater than endlrsn is discarded. If the endlrsn RBA value does not match the start of a log record, Db2 restart fails. If you omit ENDLRSN, Db2 determines the end of the log range.

On the CHECKPT statement, endlrsn is the LRSN of the end checkpoint log record.

STARTIME=startime

Enables you to record the start time of the RBA in the BSDS. This field is optional.

startime specifies the start time in the following timestamp format:

```
yyyydddhhmmst
```

In this format:

- `yyyy` Indicates the year (1989-2099).
- `ddd` Indicates the day of the year (0-365; 366 in leap years).
- `hh` Indicates the hour (0-23).
- `mm` Indicates the minutes (0-59).
- `ss` Indicates the seconds (0-59).
- `t` Indicates tenths of a second.

If fewer than 14 digits are specified for the STARTIME or ENDTIME parameter, trailing zeros are added.

If STARTIME is specified, the ENDTIME, STARTRBA, and ENDRBA options must also be specified.
**ENDTIME= endtime**

Enables you to record the end time of the RBA in the BSDS. This field is optional.

endtime specifies the end time in the same timestamp format as the STARTTIME option. The ENDTIME value must be greater than or equal to the value of STARTTIME.

**DELETE**

Deletes either CCSID information or log data set information from the bootstrap data sets. To delete CCSID information, specify the CCSIDS option. To delete all information for a specified log data set or volume, specify the DSNAMES option.

**CCSIDS**

Deletes CCSID information from the BSDS. CCSID information is stored in the BSDS to ensure that you do not accidentally change the CCSID values.

Use this option under the direction of IBM Support when the CCSID information in the BSDS is incorrect. After you run a DSNJU003 job with the DELETE CCSIDS option, the CCSID values from the application defaults load module are recorded in the BSDS the next time Db2 is started.

**CRESTART**

Controls the next restart of Db2, either by creating a new conditional restart control record or by canceling the one that is currently active.

**CREATE**

Creates a new conditional restart control record. When the new record is created, the previous control record becomes inactive.

**SYSPITR=log-truncation-point**

Specifies the log RBA (non-data sharing system) or the log LRSN (data sharing system) that represents the log truncation point for the point-in-time for system recovery. Before you run the RESTORE SYSTEM utility to recover system data, you must use the SYSPITR option of DSNJU003. This option enables you to create a conditional restart control record to truncate the logs for system point-in-time recovery. You can also specify a value of FFFFFFFFFFFFFFFFFFFFFF to cause a point-in-time recovery to occur without log truncation.

log-truncation-point specifies the log RBA, log LRSN, or log FFFFFFFFFFFFFFFFFFFFF. In a non-data sharing environment, log-truncation point is the RBA value that matches the start of the last log record that is to be used during restart. If the RBA value does not match the start of a log record, Db2 restart fails. In a data sharing environment, log-truncation point is an LRSN value that is a valid log truncation point. A valid log truncation point is any LRSN value for which there exists a log record with an LRSN that is greater than or equal to the specified LRSN value. Use the same LRSN value for all members of the data sharing group that require log truncation.

You cannot specify any other option with CREATE, SYSPITR.

**ENDTIME= log-truncation-timestamp**

Specifies an end time value that is to be used as the log truncation point. A valid truncation point is any UTC timestamp for which there exists a log record with a timestamp that is greater than or equal to the specified timestamp value. Any log information in the bootstrap data set, the active logs, and the archive logs with a timestamp greater than the ENDTIME is discarded.

If you do not specify ENDTIME, Db2 determines the end of the log range.

You cannot specify any other option with CREATE, ENDTIME.
SYSPIRT= log-truncation-timestamp

Specifies the timestamp value that represents the point-in-time log truncation point for system recovery. Before you run the RESTORE SYSTEM utility to recover system data, you must use the SYSPITR or SYSPITRT option of DSNJU003. The options enable you to create a conditional restart control record to truncate the logs for system point-in-time recovery.

Log-truncation-timestamp specifies a timestamp value that is to be used as the log truncation point. A valid log truncation point is any UTC timestamp for which there exists a log record with a timestamp that is greater than or equal to the specified timestamp value. Any log information in the bootstrap data set, the active logs, and the archive logs with a timestamp greater than SYSPITRT is discarded. If you omit SYSPITR, Db2 determined the end of the log range. Use the same timestamp value for all members of the data sharing group that require log truncation.

You cannot specify any other option with CREATE, SYSPITRT.

Note: The startime keyword specifies the start time in the yyyydddhhmmsst timestamp format. See the STARTIME option for details about the timestamp.

CANCEL

On the CRESTART statement, deactivates the currently active conditional restart control record. The record remains in the BSDS as historical information.

No other keyword can be used with CANCEL on the CRESTART statement.

On the CHECKPT statement, deletes the checkpoint queue entry that contains a starting RBA that matches the parameter that is specified by the STARTRBA keyword.

Attention: This statement can override DB2’s efforts to maintain data in a consistent state. Do not use this statement without understanding the conditional restart process.

CHKPTRBA= chkptrba

Identifies the log RBA of the start of the checkpoint record that is to be used during restart.

If you use STARTRBA or ENDRBA, and you do not use CHKPTRBA, the DSNJU003 utility selects the RBA of an appropriate checkpoint record. If you do use CHKPTRBA, you override the value that is selected by the utility.

chkptrba must be in the range that is determined by startrba and endrba or their default values.

If possible, do not use CHKPTRBA; let the utility determine the RBA of the checkpoint record.

CHKPTRBA=0 overrides any selection by the utility; at restart, Db2 attempts to use the most recent checkpoint record.

FORWARD=

Indicates whether to use the forward-log-recovery phase of Db2 restart, which reads the log in a forward direction to recover any units of recovery that were in one of the following two states when Db2 was last stopped:

- Indoubt (the units of recovery had finished the first phase of commit, but had not started the second phase)
- In-commit (had started but had not finished the second phase of commit)

YES

Allows forward-log recovery.
If you specify a cold start (by using the same value for STARTRBA and ENDRBA), no recovery processing is performed.

**NO** Terminates forward-log recovery before log records are processed. When you specify, FORWARD=NO, Db2 does not go back in the log to the beginning of any indoubt or in-commit units of recovery to complete forward recovery for these units. Choose this option if a very old indoubt unit of recovery exists to avoid a lengthy restart. The in-commit and indoubt units of recovery are marked as bypassed and complete in the log. However, any database writes that are pending at the end of the log, including updates from other units of recovery, are still written out during the forward phase of restart. Any updates that must be rolled-back, such as for an inflight or in-abort unit of recovery, are done during the backout phase of restart.

**BACKOUT**
- Indicates whether to use the backward-log-recovery phase of Db2 restart, which rolls back any units of recovery that were in one of the following two states when Db2 was last stopped:
  - Inflight (did not complete the first phase of commit)
  - In-abort (had started but not finished an abort)
- **YES** Allows backward-log recovery.
  - If you specify a cold start (by using the same value for STARTRBA and ENDRBA), no recovery processing is performed.
- **NO** Terminates backward-log recovery before log records are processed.

**CSRONLY**
- Performs only the first and second phases of restart processing (log initialization and current-status rebuild). After these phases, the system status is displayed, and restart terminates. Some parts of the log initialization are not performed, including any updating of the log and display of STARTRBA and ENDRBA information.
- When Db2 is restarted with this option in effect, the conditional restart control record is not deactivated. To prevent the control record from remaining active, use the DSNJU003 utility again with CRESTART CANCEL, or with CRESTART CREATE to create a new active control record.

**NEWCAT**
- Changes the VSAM catalog name in the BSDS.

**VSAMCAT= catalog-name**
- Changes the VSAM catalog name entry in the BSDS.
  - *catalog-name* can be up to eight characters long. The first character must be alphabetic, and the remaining characters can be alphanumeric.

**DDF**
- Updates the LOCATION, LUNAME, and other DDF related information values in the BSDS. If you use this statement to insert new values into the BSDS, you must include at least the LOCATION in the DDF statement. To update an existing set of values, you need to include only those values that you want to change. The DDF record cannot be deleted from the BSDS after it has been added; it can only be modified.

**LOCATION= location-name**
- Changes the LOCATION value in the BSDS.
location-name specifies the name of your local Db2 site.

**PORT**

Identifies the TCP/IP port number that is used by DDF to accept incoming connection requests. This value must be a decimal number between 0 and 65535, including 65535; 0 indicates that DDF TCP/IP support is to be deactivated. Set PORT to 0 if DDF needs to use only VTAM®, and will not use TCP/IP.

If Db2 is part of a data sharing group, all the members of the Db2 data sharing group must have the same value for PORT.

**RESPORT**

Identifies the TCP/IP port number that is used by DDF to accept incoming DRDA two-phase commit resynchronization requests. This value must be a decimal number between 0 and 65535, including 65535; zero indicates that DDF’s TCP/IP support is to be deactivated. If RESPOR is non-zero, RESPOR must not be the same as the value that is supplied on PORT.

For data sharing Db2 systems, RESPOR must be uniquely assigned to each Db2 member, so that no two Db2 members use the same TCP/IP port for two-phase commit resynchronization.

**SECPOR**

Identifies the TCP/IP port number that is used by DDF to accept inbound secure DRDA connection requests. This value must be a decimal number between 0 and 65535, including 65535; zero indicates that DDF’s secure connection support for TCP/IP is deactivated.

**ALIAS= alias-name :alias-port :alias-secport**

Specifies one or more alias names for the location. An alias name is a name besides the location name that connect processing can accept. Specifying an alias name does not change the location identifier for a database object.

**Important:** ALIAS applies to DRDA connections only.

You can specify and modify as many as eight location aliases by using the DSNJU003 utility. To specify more than 8 aliases, use the MODIFY DDF command with the ALIAS option.

*alias-name* specifies from 1 to 16 characters for the location name. *alias-name* cannot be one of the valid DSNJU003 keywords.

* :alias-port* specifies a TCP/IP port number for the alias that can be used by DDF to accept distributed requests. This value must be a decimal number between 1 and 65535, including 65535. The value must be different from the values for the PORT, RESPOR, and SECPOR options and any value that was specified for *alias-port or alias-secport* of any other defined alias. Specify a value for *alias-port* when you want to identify a subset of data sharing members to which a distributed request can go.

* :alias-secport* specifies a secure TCP/IP port number for the alias that can be used by DDF to accept secure distributed requests using SSL. This value must be a decimal number between 1 and 65535, including 65535. The value must be different from the values for the SECPOR, PORT, and RESPOR options, and any value that was specified for *alias-port or alias-secport* of any other defined alias. Specify a value for *alias-secport* when you want to identify a subset of data sharing members to which a secure distributed request can go.

You can add or replace aliases by respecifying the ALIAS option. The new list of names replaces the existing list.
**Related information:**

- [MODIFY DDF (Db2) (Db2 Commands)]

**IPNAME=ipname**

Identifies and associates an IPNAME value with DDF.

*ipname* can be up to 8 bytes in length and must be an alphanumeric string, beginning with a letter. When you specify this option, DDF activates only its TCP/IP communications support, regardless of whether or not there is a value for LUNAME. Only inbound and outbound DRDA protocol communications over TCP/IP are allowed. Db2 makes no attempt to activate SNA/APPC communications support.

The value specified must be either unique to this Db2 subsystem within an enterprise, or if the Db2 subsystem is configured to be a member of a data sharing group, unique to the data sharing group for which this Db2 subsystem is a member. All members of a data sharing group must be defined with the same IPNAME value if all the members are to activate only their TCP/IP communications support. If some members of a data sharing group activate their SNA/APPC (as well as TCP/IP communications support), then the IPNAME value chosen for the TCP/IP only members must match the GENERIC value specified for the members which activate their SNA/APPC as well as their TCP/IP communications support.

Db2 uses the IPNAME value as the network-ID portion of a unit-of-work identifier. A unit-of-work identifier has traditionally been made up of a network-ID, an LU name, a 6-byte unique identifier created from a timestamp, and a 2-byte current commit count value. When running with an IPNAME value, the LU name portion of the unit-of-work identifier is created from a 4-byte character representation of the hexadecimal notation of the value specified for the TCP/IP resync port (RESPORT).

Do not confuse IPNAME with a TCP/IP external such as a hostname or domain name. The value you give IPNAME is only used internally by Db2 or in a DRDA exchange with another DRDA server. It cannot be referenced by any TCP/IP external, such as ping. Also, the hostname of the system upon which the Db2 is running is a poor choice for IPNAME because more than one Db2 could run on the same system, and the IPNAME value given to any Db2 or Db2 data sharing group must be unique within an enterprise.

**IPV4= ipv4-address**

Identifies and associates a constant IPv4 IP address with DDF to accept incoming connection requests to this specific subsystem only. This address must be entered in dotted decimal form. If an IP address is not specified, Db2 will automatically determine the IP address from TCP/IP.

When Db2 is a member of a data sharing group, it is strongly recommended that you refer to a dynamic virtual IP address (DVIP). A group IP address, GRPIPV4, should also be specified.

**IPV6= ipv6-address**

Identifies and associates a constant IPv6 IP address with DDF to accept incoming connection requests to this specific subsystem only. This address must be entered in colon hexadecimal form. If an IP address is not specified, Db2 will automatically determine the IP address from TCP/IP.

When Db2 is a member of a data sharing group, it is strongly recommended that you refer to a dynamic virtual IP address (DVIP). A group IP address, GRPIPV6, should also be specified.
GRPIPV4
Identifies and associates a constant IPv4 IP address with the data sharing group for which this DDF is a member. The IP address is used to accept incoming connection requests that can be serviced by any member of the data sharing group. This address must be entered in dotted decimal form. An associated IPv4 subsystem/member address must also be specified in order to identify the IP address associated with this specific member of the group. If an IP address is not specified, Db2 will automatically determine the IP address from TCP/IP.

It is strongly recommended that you refer to a sysplex distributor owned distributing dynamic virtual IP address (DVIPA).

GRPIPV6
Identifies and associates a constant IPv6 IP address with the data sharing group for which this DDF is a member. The IP address is used to accept incoming connection requests that can be serviced by any member of the data sharing group. This address must be entered in colon hexadecimal form. An associated IPv6 subsystem/member address must also be specified in order to identify the IP address associated to this specific member of the group. If an IP address is not specified, Db2 will automatically determine the IP address from TCP/IP.

It is strongly recommended that you refer to a sysplex distributor owned distributing dynamic virtual IP address (DVIPA).

LUNAME= luname
Changes the LUNAME value in the BSDS.

luname specifies the LUNAME value. The LUNAME in the BSDS must always contain the value that identifies your local Db2 subsystem to the VTAM network.

PASSWORD=
The DDF password follows VTAM convention, but Db2 restricts it to one to eight alphanumeric characters. The first character must be either a capital letter or an alphabetic extender. The remaining characters can consist of alphanumeric characters and alphabetic extenders.

password
Optionally assigns a password to the distributed data facility communication record that establishes communications for a distributed data environment. The PRTCT=password option on the APPL definition statement is used to define Db2 to VTAM.

GENERIC= gluname
Replaces the value of the Db2 GENERIC LUNAME subsystem parameter in the BSDS.

gluname specifies the GENERIC LUNAME value.

NOPASSWD
Removes the archive password protection for all archives that are created after this operation. It also removes a previously existing password from the DDF record. No other keyword can be used with NOPASSWD.

NGENERIC
Changes the Db2 GENERIC LUNAME to binary zeros in the BSDS, indicating that no VTAM generic LU name support is requested.
NOALIAS
Indicates that no alias names exist for the specified location. Any alias names that were specified in a previous DSNJU003 utility job are removed.

NOIPV4
Removes the constant IPv4 address from the BSDS. The NGRPIPV4 keyword must also be specified to ensure that the associated group address, if any, is also removed.

NOIPV6
Removes the constant IPv6 address from the BSDS. The NGRPIPV6 keyword must also be specified to ensure that the associated group address, if any, is also removed.

NGRPIPV4
Removes the constant data sharing group IPv4 address from the BSDS.

NGRPIPV6
Removes the constant data sharing group IPv6 address from the BSDS.

NOIPNAME
Removes the IPNAME value from the DDF record. No other keyword can be used with NOIPNAME.

NOLUNAME
Removes the LUNAME value from the DDF record. No other keyword can be used with NOLUNAME.

CHECKPT
Allows updating of the checkpoint queue with the start checkpoint and end checkpoint log records.

Attention: This statement can override DB2's efforts to maintain data in a consistent state. Do not use the statement without understanding the conditional restart and checkpoint processing processes.

TIME= time
On the CHECKPT statement, specifies the time that the start checkpoint record was written.

On the HIGHRBA statement, TIME specifies when the log record with the highest RBA was written to the log.

time specifies the time value. For timestamp format, see the STARTIME option description.

HIGHRBA
Updates the highest-written log RBA in either the active or archive log data sets.

Attention: This statement can override DB2's efforts to maintain data in a consistent state. Do not use the statement without understanding the conditional restart process.

OFFLRBA= offlrba
Specifies the highest-offloaded RBA in the archive log.

offlrba is a hexadecimal number of up to 20 characters. If you use fewer than 20 characters, leading zeros are added. The value must end with hexadecimal X'FFF'.

DELMBR
Deactivates or destroys a member of a data sharing group.
**DEACTIV**
Marks a member of a data sharing group for deactivation. Deactivation is the first step in deletion of a member from a data sharing group.

Before the member can be deactivated, it must be quiesced and have no outstanding work. The logs and BSDS must exist.

**DESTROY**
Completes the deletion of a member from a data sharing group.

After a member is destroyed, its member ID can be reused, and the logs and BSDS can be deleted.

**RSTMBR**
Restores a deactivated member of a data sharing group to the quiesced state.

**MEMBERID= member-id**
Specifies the data sharing group member that is to be deactivated, destroyed, or restored.

*member-id* is a number in the range 1 - 32. This number is the member ID that shown in the output from the DISPLAY GROUP command or the DSNJU004 (print log map) utility.

**Related information:**
- "DSNJU004 (print log map) output" on page 929
- DSN7100I (Db2 Messages)

**Related concepts:**
- Member-specific access (Db2 Data Sharing Planning and Administration)
- Phase 3: Forward log recovery (Db2 Administration Guide)
- Timestamp (Db2 SQL)

**Related tasks:**
- Deleting data sharing members (Db2 Data Sharing Planning and Administration)
- Performing conditional restart (Db2 Administration Guide)
- Restoring deactivated data sharing members (Db2 Data Sharing Planning and Administration)

**Related information:**
- PRTCT (SNA Resource Definition Reference)

---

**Making changes for active logs**

You can add, delete, record, enlarge, and encrypt active logs.

**Adding active logs**

If an active log is in stopped status, it is not reused for output logging; however, it continues to be used for reading. To add a new active log:

1. Use the Access Method Services DEFINE command to define new active log data sets.
2. Use DSNJLOGF to preformat the new active log data sets.
If you do not preformat these logs with the DSNJLOGF utility, Db2 needs to preformat them the first time that they are used, and performance might be impacted. Empty data sets and data sets with residual data need to be preformatted.

3. Use DSNJU003 to register the new data sets in the BSDS.

For example, specify the following statements:

```
NEWLOG DSNAME=DSNC110.LOGCOPY1.DS04,COPY1
NEWLOG DSNAME=DSNC110.LOGCOPY2.DS04,COPY2
```

To copy the contents of an old active log data set to the new one, you can also give the RBA range and the starting and ending timestamp on the NEWLOG statement.

To archive to disk when the size of your active logs has increased, you might find it necessary to increase the size of your archive log data set primary and secondary space quantities by increasing the values of subsystem parameters PRIQTY and SECQTY.

### Deleting active logs

To delete information about an active log data set from the BSDS, you might specify the following statements:

```
DELETE DSNAME=DSNC110.LOGCOPY1.DS01
DELETE DSNAME=DSNC110.LOGCOPY2.DS01
```

### Recording active logs

To record information about an existing active log data set in the BSDS, you might specify the following statement:

```
NEWLOG DSNAME=DSNC110.LOGCOPY2.DS05,COPY2,STARTIME=19910212205198,
        ENDTIME=19910412205200,STARTRBA=43F8000,ENDRBA=65F3FFF
```

You can insert a record of that information into the BSDS for any of these reasons:
- The data set has been deleted and is needed again.
- You are copying the contents of one active log data set to another data set (copy 1 to copy 2).
- You are recovering the BSDS from a backup copy.

### Enlarging active logs

When Db2 is inactive (down), use one of the following procedures.

If you can use the Access Method Services REPRO command, follow these steps:

1. Stop Db2. This step is required because Db2 allocates all active log data sets when it is active.
2. Use the Access Method Services ALTER command with the NEWNAME option to rename your active log data sets.
3. Use the Access Method Services DEFINE command to define larger active log data sets. Refer to installation job DSNTIJIN to see the definitions that create the original active log data sets.

By reusing the old data set names, you don't need to run the change log inventory utility to establish new names in the BSDSs. The old data set names and the correct RBA ranges are already in the BSDSs.
4. Use the Access Method Services REPRO command to copy the old (renamed) data sets into their respective new data sets.

5. Start Db2.

If you cannot use the Access Method Services REPRO command, follow this procedure:

1. Ensure that all active log data sets except the current active log data sets have been archived. Active log data sets that have been archived are marked REUSABLE in print log map utility (DSNJU004) output.

2. Stop Db2.

3. Rename or delete the reusable active logs. Allocate new, larger active log data sets with the same names as the old active log data sets.

4. Run the DSNJLOGF utility to preformat the new log data sets.

5. Run the change log inventory utility (DSNJU003) with the DELETE statement to delete all active logs except the current active logs from the BSDS.

6. Run the change log inventory utility with the NEWLOG statement to add to the BSDS the active logs that you just deleted. So that the logs are added as empty, do not specify an RBA range.

7. Start Db2.

8. Issue the ARCHIVE LOG command to cause Db2 to truncate the current active logs and switch to one of the new sets of active logs.

9. Repeat steps 2 through 7 to enlarge the active logs that were just archived.

Although all log data sets do not need to be the same size, from an operational standpoint using the same size is more consistent and efficient. If the log data sets are not the same size, tracking your system’s logs can be more difficult. Space can be wasted if you are using dual data sets of different sizes because they fill only to the size of the smallest, not using the remaining space on the larger one.

If you are archiving to disk and the size of your active logs has increased, you might need to increase the size of your archive log data sets. However, because of DFSMS disk management limits, you must specify less than 64,000 tracks for the primary space quantity.

Related tasks:

- Encrypting log, catalog, and directory data sets with z/OS DFSMS data set encryption (Managing Security)

Related reference:

- PRIMARY QUANTITY field (PRIQTY subsystem parameter) (Db2 Installation and Migration)
- SECONDARY QTY field (SECQTY subsystem parameter) (Db2 Installation and Migration)

### Making changes for archive logs

You can add and delete archive logs.

**Adding:** When the recovery of an object depends on reading an existing archive log data set, the BSDS must contain information about that data set, so that the recovery job can find it. To register information about an existing archive log data set in the BSDS, you might specify the following statement:
Deleting: To delete an entire archive log data set from one or more volumes, you might specify the following statement:

```
DELETE DSNNAME=DSNC110.ARCHLOG1.D89021.T2205197.A0000015,COPY1VOL=DSNV04
```

A conditional restart control record

You can create a conditional restart control record in the BSDS.

To create a new conditional restart control record in the BSDS, you must execute the change log inventory utility and use the CRESTART control statement. For example, to truncate the log, to specify the earliest log RBA, and to bypass backout, use a statement similar to the following statement:

```
CRESTART CREATE,STARTTRBA=28000,ENDRBA=58000,BACKOUT=NO
```

To specify a cold start, make the values of STARTTRBA and ENDRBA equal with a statement similar to the following statement:

```
CRESTART CREATE,STARTTRBA=4A000,ENDRBA=4A000
```

In most cases when doing a cold start, you should make sure that the STARTTRBA and ENDRBA are set to an RBA value that is greater than the highest used RBA.

To truncate the Db2 logs via conditional restart by specifying a timestamp rather than an RBA value, use a statement similar to the following statement:

```
CRESTART CREATE,ENDTIME=20051402030068
```

An existing conditional restart control record governs any START Db2 operation until one of these events occurs:
- A restart operation completes.
- A CRESTART CANCEL statement is issued.
- A new conditional restart control record is created.

Deleting log data sets with errors

If an active log data set encounters an I/O error, use the DSNJU003 (change log inventory) utility to delete the log data sets with errors.

Procedure

To delete log data sets with errors:

1. If you use dual active log data sets, check if the data from the bad active log data set is saved in the other active log. If it is, you can use the other active log.
2. If you cannot use the other active log or if the active log is in the STOPPED status, fix the problem manually by taking the following steps:
   a. Check whether the data set was offloaded. For example, check the list of archive log data sets to see whether one has the same RBA range as the active log data set. This list can be created by using the DSNJU004 (print log map) utility.
   b. If the data set was not offloaded, copy the data to a new VSAM data set. If the data set was offloaded, create a new VSAM data set that is to be used as an active log data set.

Chapter 40. DSNJU003 (change log inventory) 921
c. Run the change log inventory utility with the DELETE and NEWLOG statements.

**Important:** If misused, the change log inventory utility can compromise the viability and integrity of the Db2 subsystem. Only highly skilled people, such as the Db2 system administrator, should use this utility, and then only after careful consideration.

The DELETE statement removes information about the bad data set from the BSDS. The NEWLOG statement identifies the new data set as the new active log. The DELETE and NEWLOG operations can be performed by the same job step. The DELETE statement precedes the NEWLOG statement in the SYSIN input data set.

To ensure consistent results, run the change log inventory utility on the same z/OS system on which the Db2 online subsystem runs.

Use the print log map utility before and after you run the change log inventory utility to ensure correct execution and to document changes.

When you use dual active logs, choose a naming convention that distinguishes primary and secondary active log data set. The naming convention should also identify the log data sets within the series of primary or secondary active log data sets. For example, the default naming convention that is established at Db2 installation time is as follows:

\[ \text{prefix}.\text{LOGCOPY}n.DSmn \]

In this convention, \(n=1\) for all primary log data sets, \(n=2\) for all secondary log data sets, and \(mm\) is the data set number within each series.

If a naming convention such as the default convention is used, pairs of data sets with equal \(mm\) values are usually used together. For example, \(\text{prefix}.\text{LOGCOPY1.DS02}\) and \(\text{prefix}.\text{LOGCOPY2.DS02}\) are used together.

However, after you run the change log inventory utility with the DELETE and NEWLOG statements, the primary and secondary series can become unsynchronized. This situation can occur even if the NEWLOG data set name that you specify is the same as the old data set name. To avoid this situation, always do maintenance on both data sets of a pair in the same change log inventory execution:
- Delete both data sets together.
- Define both data sets together with NEWLOG statements.

The data sets themselves do not require deletion and redefinition.

3. Delete the bad data set by using VSAM Access Method Services.

**What to do next**

Before you initiate a conditional restart or cold restart, consider making backup copies of all disk volumes that contain any Db2 data sets. These backup copies enable a possible fallback. The backup data sets must be generated when Db2 is not active.

**Related reference:**
- Chapter 41, “DSNJU004 (print log map),” on page 927
- Chapter 40, “DSNJU003 (change log inventory),” on page 903

**Altering references to log data sets in the BSDS**

You can add or delete active or archive log data sets in the bootstrap data set (BSDS) by using the DSNJU003 utility.
About this task

Introductory concepts

Bootstrap data set (Introduction to Db2 for z/OS)

When you alter references to log data sets in the BSDS, the log data sets are not changed. And you do not need to make any changes to the referenced log data sets.

Procedure

To alter references to log data sets in the BSDS:

- To add a reference to a data set in the BSDS, use the NEWLOG statement of the DSNJU003 utility.
- To delete a reference to a data set in the BSDS, use the DELETE statement of the DSNJU003 utility.

Related reference:

Chapter 40, “DSNJU003 (change log inventory),” on page 903

Defining the high-level qualifier for catalog and directory objects

You can define the high-level qualifier for catalog and directory objects.

Procedure

Use the NEWCAT statement to define the high-level qualifier that is to be used for the following objects:

- Catalog table spaces and index spaces
- Directory table spaces and index spaces

At startup, the Db2 system checks that the name that is recorded with NEWCAT in the BSDS is the high-level qualifier of the Db2 system table spaces that are defined in the load module for subsystem parameters.

NEWCAT is normally used only at installation time.

When you change the high-level qualifier by using the NEWCAT statement, you might specify the following statements:

```
//S2 EXEC PGM=DSNJU003  
//SYSUT1 DD DSN=DSNCl20.BSDS01,DISP=OLD  
//SYSUT2 DD DSN=DSNCl20.BSDS02,DISP=OLD  
//SYSPRINT DD SYSOUT=*
   NEWCAT VSAMCAT=DBP1
```

After you run the change log inventory utility with the NEWCAT statement, the utility generates output similar to the following output:

```
NEWCAT VSAMCAT=DBP1
DSNJ210I OLD VASAM CATALOG NAME=DSNCl20, NEW CATALOG NAME=DBP1
DSNJ225I NEWCAT OPERATION COMPLETED SUCCESSFULLY
DSNJ200I DSNJU003 CHANGE LOG INVENTORY UTILITY PROCESSING COMPLETED SUCCESSFULLY
```

Related tasks:

“Renaming Db2 system data sets”

Renaming Db2 system data sets

Occasionally, you might want to rename the Db2 system table spaces
**Procedure**

To rename Db2 system data sets:

1. Stop Db2 in a consistent state.
2. Create a full system backup so that you can recover from operational errors.
3. Execute the change log inventory utility with NEWCAT.
4. Rename the BSDS and all Db2 directory and catalog table spaces and index spaces with IDCAMS.
5. Reassemble DSNZPARM to redefine the high-level qualifier for the system table spaces.
6. Update the BSDS name in the Db2 startup procedure.
7. Start Db2.
8. Drop and re-create the work file database.
9. Optionally use the ALTER command for table spaces in DSNDB04 and user databases.

---

**Renaming Db2 active log data sets**

When you rename system data sets, you might also want to rename the log data sets.

**About this task**

To rename Db2 active log data sets:

**Procedure**

1. Stop Db2 in a consistent state.
2. Create a full system backup so that you can recover from operational errors.
3. Delete the reusable active log data sets with IDCAMS, but keep the current active log.
4. Define a new set of active log data sets with IDCAMS.
5. Execute the change log inventory utility to remove names of deleted active log data sets and to define the new active log data set names in the BSDS.
6. Start and use Db2 normally.

**Results**

When the current active log is archived and becomes reusable, you can delete it.

---

**Renaming Db2 archive log data sets**

You do not need to rename archive log data sets because old archive logs are replaced as a part of the normal maintenance cycle and the RECOVER utility works with archive logs that contain different high-level qualifiers.

To modify the high-level qualifier for archive log data sets, you need to reassemble DSNZPARM.

---

**Sample DSNJU003 control statements**

Use the sample control statements as models for developing your own DSNJU003 control statements.
Example 1: Adding a new archive log data set

The following control statement specifies that the DSNJU003 utility is to add the data set DSNREP.A001187 to the BSDS. The volume serial number for the data set is DSNV04, as indicated by the COPY1VOL option. The device type is SYSDA, and the data set is not to be cataloged. The RBA of the beginning of the archive log data set volume is 3A190000, and the end RBA is 3A1F0FFF.

//STEP5 EXEC PGM=DSNJU003,COND=EVEN
//SYSUT1 DD DSN=DSNCAT.BSDS01,DISP=SHR
//SYSUT2 DD DSN=DSNCAT.BSDS02,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
NEWLOG DSNAME=DSNREPAL.A0001187,COPY1VOL=DSNV04,UNIT=SYSDA,
STARTRBA=3A190000,ENDRBA=3A1F0FFF,CATALOG=NO
*/

Example 2: Deleting a data set

The following control statement specifies that DSNJU003 is to delete data set DSNREP.A0001187 from the BSDS. The volume serial number for the data set is DSNV04, as indicated by the COPY1VOL option.

DELETE DSNAME=DSNREPAL.A0001187,COPY1VOL=DSNV04

Example 3: Creating a new conditional restart control record

The following statement specifies that DSNJU003 is to create a new conditional restart control record, which controls the next restart of Db2. BACKOUT=NO indicates that Db2 is not to execute the backward-log-recovery phase when it restarts. The ENDRBA option indicates that 00000010000 is the last RBA of the log that is to be used during restart. Any log information in the bootstrap data set, the active logs, and the archive logs with an RBA that is greater than this RBA is discarded.

CRESTART CREATE,BACKOUT=NO,ENDRBA=00000010000

Example 4: Adding a communication record to the BSDS

The following control statement specifies that DSNJU003 is to add a new communication record to the BSDS. The location, LU name, and password values are all provided.

DDF LOCATION=USIBMSTODB22,LUNAME=STL#M08,PASSWORD=$STL@290

Example 5: Updating a communication record with a secure TCP/IP port number in the BSDS

The following control statement specifies that DSNJU003 is to update the communication record in the BSDS to specify a secure TCP/IP port.

DDF LOCATION=XYZ,SECPORT=448

Example 6: Adding a communication record with an alias to the BSDS

The following control statement specifies that DSNJU003 is to add a communication record to the BSDS. The location, alias, LU name, and password values are all provided.

DDF LOCATION=USIBMSTODB22,ALIAS=STL715A1,STL715A2,LUNAME=STL#M08,PASSWORD=$STL@290
Note: The alias is an SQL identifier and should follow the rules of SQL identifiers. The identifier can not include special characters when you are naming a location alias.

Example 7: Adding multiple aliases and alias ports to the BSDS

The following control statement specifies five alias names for the communication record in the BSDS (MYALIAS1, MYALIAS2, MYALIAS3, MYALIAS4, and MYALIAS5). Only MYALIAS2 and MYALIAS5 support subsets of a data sharing group. Any alias names that were specified in a previous DSNJU003 utility job are removed.

DDF ALIAS=MYALIAS1,MYALIAS2:8002,MYALIAS3,MYALIAS4,MYALIAS5:10001

Example 8: Specifying a point in time for system recovery

The following control statement specifies that DSNJU003 is to create a new conditional restart control record. The SYSPITR option specifies an end RBA value as the point in time for system recovery for a non-data sharing system. For a data sharing system, use an end LRSN value instead of an end RBA value. This point in time is used by the RESTORE SYSTEM utility.

//JOBLIB DD DSN=USER.TESTLIB,DISP=SHR
// DD DSN=DSN910.SDSNLOAD,DISP=SHR
//STEP01 EXEC PGM=DSNJU003
//SYSTU1 DD DSN=DSNC910.BSDS01,DISP=OLD
//SYSTU2 DD DSN=DSNC910.BSDS02,DISP=OLD
//SYSPRINT DD SYSUT2=
//SYSIN DD *
CRESTART CREATE,SYSPITR=04891665D000
/*

To indicate that a SYSPITR restart is to be done without log truncation, specify a SYSPITR value of all 'FFs in the DSNJU003 job with the CRESTART parameter.

//STEP1 EXEC PGM=DSNJU003
//SYSTU1 DD DSN=DSNC910.BSDS01,DISP=OLD
//SYSTU2 DD DSN=DSNC910.BSDS02,DISP=OLD
//SYSPRINT DD SYSUT2=
//SYSIN DD *
CRESTART CREATE,SYSPITR=FFFFFFFFFFFFFFFFFFFF
/*

During the subsequent restart, the user will be asked to confirm the conditional restart with the following:

DSNJ256I ) DSNJW6 CONDITIONAL RESTART RECORD INDICATES SYSPITR
RESTART WITH NO LOG TRUNCATION
DSNJ11I ) CONDITIONAL RESTART RECORD 1 CREATED AT 7.214 7:56
WAS FOUND. REPLY Y TO USE, N TO CANCEL

Example 9: Removing aliases from a communication record

The following control statement specifies that no alias names apply. Any alias names that were specified in a previous DSNJU003 utility job are removed.

DDF NOALIAS

Example 10: Deactivating a data sharing member

The following control statement specifies that DSNJU003 is to deactivate a data sharing member.

DELMBR DEACTIV,MEMBERID=3
Chapter 41. DSNJU004 (print log map)

The DSNJU004 (print log map) stand-alone utility generates a variety of information that can be useful in backup and recovery situations.

The print log map (DSNJU004) utility lists the following information:
- Log data set name, log RBA association, and log LRSN for both copy 1 and copy 2 of all active and archive log data sets
- Active log data sets that are available for new log data
- Status of all conditional restart control records in the bootstrap data set
- Contents of the queue of checkpoint records in the bootstrap data set
- The communication record of the BSDS, if one exists
- Contents of the quiesce history record
- System and utility timestamps
- Contents of the checkpoint queue
- Archive log command history
- BACKUP SYSTEM utility history
- System CCSID information
- System-level backup information
- Information about deactivated and destroyed data sharing members

In a data sharing environment, the DSNJU004 utility can list information from any or all BSDSs of a data sharing group.

Environment

The DSNJU004 program runs as a batch job.

This utility can be executed either when Db2 is running and when it is not running. However, to ensure consistent results from the utility job, the utility and the Db2 online subsystem must both be executing under the control of the same operating system.

Authorization required

The user ID of the DSNJU004 job must have requisite RACF authorization.

Required and optional data sets

DSNJU004 recognizes DD statements with the following DD names:

**SYSUT1**
Specifies and allocates the bootstrap data set. This statement is required. It allocates the BSDS. If the BSDS must be shared with a concurrently executing Db2 online subsystem, use DISP=SHR on the DD statement.

**SYSPRINT**
Specifies a data set or print spool class for print output. This statement is required. The logical record length (LRECL) is 125.

**SYSIN (optional)**
Contains the control statement. If you do not specify the SYSIN DD statement, BSDS information is printed only from the BSDS data set that is identified by the SYSUT1 DD statement.
GROUP
Names a single BSDS. Db2 can use this BSDS to find the names of all BSDSs in the group. Ensure that the BSDS name that you specify is not the BSDS of a member that has been quiesced since before new members joined the group. This statement is required if the control statement specifies either of these options:
- MEMBER *
- MEMBER(member-name)

MnnBSDS
Names the BSDS data set of a group member whose information is to be listed. You must specify one such DD statement for each member. The statements are required if the control statement specifies MEMBER DDNAME. mnn represents a two-digit number. You must use consecutive two-digit numbers from 01 to the total number of required members. If a break occurs in the sequence of numbers, any number after the break is ignored.

Running the DSNJU004 utility

Use the following EXEC statement to execute this utility:
```
// EXEC PGM=DSNJU004
```

Recommendations

- For dual BSDSs, execute the print log map utility twice, once for each BSDS, to compare their contents.
- To ensure consistent results for this utility, execute the utility job on the same z/OS system on which the Db2 online subsystem executes.
- Execute the print log map utility regularly, possibly daily, to keep a record of recovery log data set usage.
- Use the print log map utility to document changes that are made by the change log inventory utility.

Related concepts:
- Management of the bootstrap data set (Db2 Administration Guide)
- Conditional restart (Db2 Administration Guide)

Related tasks:
- Deleting data sharing members (Db2 Data Sharing Planning and Administration)
- Restoring deactivated data sharing members (Db2 Data Sharing Planning and Administration)

Syntax and options of the DSNJU004 control statement

Using the SYSIN data set allows you to list information from any or all BSDSs of a data sharing group.
Option descriptions

The following keywords can be used in an optional control statement on the SYSIN data set:

**MEMBER**

Specifies which member's BSDS information to print.

*  Prints the information from the BSDS of each member in the data sharing group.

**DDNAME**

Prints information from only those BSDSs that are pointed to by the MxxBSDS DD statements.

**(member-name)**

Prints information for only the named group members.

Sample DSNJU004 control statement

Use the sample control statements as models for developing your own DSNJU004 control statements.

The following statement specifies that DSNJU004 is to print information from the BSDS for each member in the data sharing group:

```plaintext
//PLM EXEC PGM=DSNJU004
//GROUP DD DSN=DBD1.BSDS01,DISP=SHR
//SYSPRINT DD SYSOUT=* 
//SYSIN DD * MEMBER *
```

**DSNJU004 (print log map) output**

The output of the DSNJU004 (print log map) utility is various backup and recovery information for a subsystem or data sharing members.

Specifically, DSNJU004 output includes the following information:

- The data set name (DSN) of the BSDS.
- The system date and time (SYSTEM TIMESTAMP), and the date and time that the BSDS was last changed by the change log inventory utility (UTILITY TIMESTAMP). For more information about these timestamp values in the output, see "Timestamps in the BSDS" on page 937.
- The integrated catalog facility (ICF) catalog name that is associated with the BSDS.
- The highest-written RBA. The value is updated each time the log buffers are physically written to disk.
- The highest RBA that was offloaded.
- Log RBA ranges (STARTRBA and ENDRBA) and data set information for active and archive log data sets. The last active log data set that is listed in the output is the current active log.
- Information about each active log data set. This information includes the starting and ending RBAs within the data set, the date and time the data set was created, and the data set name (DSN) and status. You might see consecutive active or archive log data sets with an end LRSN value that is the same as the beginning LRSN value of the next data set. For a description of the active log data set information in the output, see "Active log data set status" on page 939.
- Information about each archive log data set. This information includes the starting and ending RBAs within the data set, the date and time the data set was created, and the data set name (DSN), unit and volume of storage, and status. You might see consecutive active or archive log data sets with an end LRSN value that is the same as the beginning LRSN value of the next data set.
- Conditional restart control records. For a description of this output, see "Reading conditional restart control records" on page 940.
- The contents of the checkpoint description queue. For a description of this output, see Figure 96 on page 941.
- Archive log command history. For a description of this output, see "Archive log command history" on page 940.
- The distributed data facility (DDF) communication record. This record contains the location name as defined by Db2, any alias names for the location name, and the LU name as defined by VTAM. Db2 uses this information to establish the distributed database environment.
- The tokens for all BACKUP SYSTEM utility records. The token identifies each backup version that was created. For a description of this output, see "System-level backup information" on page 942.
- The ENFM START RBA/LRSN field contains one of the following values:
  - In a non-data sharing environment, the RBA when the most recent enabling-new-function mode job started on the subsystem. If the enabling-new-function mode job was never run on the subsystem, this field contains zeroes.
  - In a data sharing environment, the LRSN when the most recent enabling-new-function mode job started on a member. If the enabling-new-function mode job was never run on a member, this field contains zeroes.
- Information about members of a data sharing group, including deactivated members and destroyed members whose slots were reclaimed.

The following figures show example output from the DSNJU004 (print log map) utility.

**Example DSNJU004 output for a subsystem**

The sample print log map utility output in the following figure is for a non-data-sharing subsystem.
**DSNJCNVT CONVERSION PROGRAM HAS NOT RUN**

**DDNAME=SYSUT1**

LOG MAP OF BSDS DATA SET COPY 1, DSN=DSNC000.DB2A.BSDS01

LTIME INDICATES LOCAL TIME, ALL OTHER TIMES ARE GMT.

DATA SHARING MODE IS OFF

**SYSTEM TIMESTAMP** - DATE=2012.256  LTIME=13:50:24.23

**UTILITY TIMESTAMP** - DATE=2012.256  LTIME=11:50:58.15

VSAM CATALOG NAME=DSNC000

HIGHEST RBA WRITTEN 0000000000007FA79BC  2012.256  20:50:57.4

HIGHEST RBA OFFLOADED 0000000000007FA6AFF

RBA WHEN CONVERTED TO V4 000000000069957FF

THIS BSDS HAS MEMBER RECORDS FOR THE FOLLOWING MEMBERS:

**HOST MEMBER NAME:**

**MEMBER ID:**

**GROUP NAME:**

**BSDS COPY 1 DATA SET NAME:**

**BSDS COPY 2 DATA SET NAME:**

**ENFM START RBA/LRSN:**

** **** DISTRIBUTED DATA FACILITY **** **

**COMMUNICATION RECORD **

20:51:19 SEPTEMBER 12, 2012

**LOCATION=STLEC1 IPNAME=(NULL) PORT=NULL SPORT=NULL RPORT=NULL**

**ALIAS=(NULL)**

**IPV4=NULL IPV6=NULL**

**GRPPV4=NULL GRPPV6=NULL**

**LUNAME=SYEC1DB2 PASSWORD=DB2PW1 GENERICCLU=(NULL)**

**ACTIVE LOG COPY 1 DATA SETS**

**START RBA/TIME**

**END RBA/TIME**

**DATE/LTIME**

**DATA SET INFORMATION**

<table>
<thead>
<tr>
<th>RBA/TIME</th>
<th>END RBA/TIME</th>
<th>DATE/LTIME</th>
<th>DATA SET INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000000000007FA68000</td>
<td>00000000000007FA6FF</td>
<td>2011.110</td>
<td>DSN=DSNC000.DB2A.LOGCOPY1.DS02 Status=REUSABLE</td>
</tr>
<tr>
<td>2012.256 20:50:44.8</td>
<td>2012.256 20:50:49.3</td>
<td>9:17</td>
<td>PASSWORD=(NULL)</td>
</tr>
<tr>
<td>0000000000007FA70000</td>
<td>00000000000007FA73FF</td>
<td>2011.110</td>
<td>DSN=DSNC000.DB2A.LOGCOPY1.DS03 Status=REUSABLE</td>
</tr>
<tr>
<td>2012.256 20:50:49.3</td>
<td>2012.256 20:50:57.4</td>
<td>9:17</td>
<td>PASSWORD=(NULL)</td>
</tr>
<tr>
<td>0000000000007FA74000</td>
<td>00000000000080E23FF</td>
<td>2011.110</td>
<td>DSN=DSNC000.DB2A.LOGCOPY1.DS01 Status=REUSABLE</td>
</tr>
<tr>
<td>2012.256 20:50:57.4</td>
<td></td>
<td>9:17</td>
<td>PASSWORD=(NULL)</td>
</tr>
</tbody>
</table>

**ARCHIVE LOG COPY 1 DATA SETS**

**NO ARCHIVE DATA SETS DEFINED FOR THIS COPY**

**ACTIVE LOG COPY 2 DATA SETS**

**NO ACTIVE DATA SETS DEFINED FOR THIS COPY**

**ARCHIVE LOG COPY 2 DATA SETS**

**NO ARCHIVE DATA SETS DEFINED FOR THIS COPY**

**CONDITIONAL RESTART CONTROL RECORD**

20:51:19 SEPTEMBER 12, 2012

** **** ACTIVE CRCC RECORD **** **

**NO CRCR RECORDS ARE ACTIVE**

** **** CRCC IDENTIFIER 0002 **** **

**USE COUNT 1**

**RECORD STATUS**

**CRCR NOT ACTIVE**

**PROCESSING STATUS**

**COLD START (STARTRBA = ENDRBA)**

**FORWARD = NO**

**BACKOUT = NO**

**STARTRBA**

000000000000000000000FE000

**ENDRBA**

000000000000000000000FE000

**ENDLRSN**

NOT SPECIFIED

**ENDTIME**

NOT SPECIFIED

**EARLIEST REQUESTED RBA**

000000000000000000000000000000000000000000

**FIRST LOG RECORD RBA**

000000000000000000000000000000000000000000

**ORIGINAL CHECKPOINT RBA**

000000000000000000000000000000000000000000

**NEW CHECKPOINT RBA (CHKPTRBA)**

NOT SPECIFIED

**CRCR CREATED**

19:51:36 SEPTEMBER 12, 2012

**BEGIN RESTART**

20:49:12 SEPTEMBER 12, 2012

**RESTART PROGRESS**

STARTED  ENDED

<table>
<thead>
<tr>
<th>CURRENT STATUS REBUILD</th>
<th>FORWARD RECOVERY PHASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO</td>
<td>NO</td>
</tr>
</tbody>
</table>

Chapter 41. DSNJU004 (print log map)  931
BACKOUT RECOVERY PHASE      NO     NO
CRCR IDENTIFIER   0001
USE COUNT        1
RECORD STATUS   CRCR NOT ACTIVE
SUCCESSFUL RESTART
PROCESSING STATUS
COLD START (STARTRBA = ENDRBA)
FORWARD = NO
BACKOUT = NO

STARTRBA       00000000000069958000
ENDRBA         00000000000069958000
ENDLRSN        NOT SPECIFIED
ENDTIME        NOT SPECIFIED
EARLIEST REQUESTED RBA 00000000000000000000
FIRST LOG RECORD RBA 00000000000000000000
ORIGINAL CHECKPOINT RBA 00000000000000000000
NEW CHECKPOINT RBA (CHKPTRBA) NOT SPECIFIED
CRCR.CREATED 16:17:06 APRIL 20, 2011
BEGIN RESTART 16:19:39 APRIL 20, 2011
END RESTART 16:19:48 APRIL 20, 2011
RESTART PROGRESS STARTED ENDED

CURRENT STATUS REBUILD   YES YES
FORWARD RECOVERY PHASE   YES YES
BACKOUT RECOVERY PHASE   YES YES
CHECKPOINT QUEUE
20:51:19 SEPTEMBER 12, 2012

TIME OF CHECKPOINT 14:10:13 JUNE 22, 2012
BEGIN CHECKPOINT RBA 00000000000007AD2C562
END CHECKPOINT RBA 00000000000007AD2ECBE
END CHECKPOINT STCK 00C9C2079978D00000000
TIME OF CHECKPOINT 17:06:49 JUNE 13, 2012
BEGIN CHECKPOINT RBA 00000000000007AD1FE4B
END CHECKPOINT RBA 00000000000007AD2B2BE
END CHECKPOINT STCK 00C9B6E530C48000000
SHUTDOWN CHECKPOINT
BEGIN CHECKPOINT RBA 00000000000007ACFBB93
END CHECKPOINT RBA 00000000000007AC6D6B4
END CHECKPOINT STCK 00C9B5E85C1961000000

REBUILD CHECKPOINT
BEGIN CHECKPOINT RBA 00000000000007ACFBB93
END CHECKPOINT RBA 00000000000007AC6D6B4
END CHECKPOINT STCK 00C9B5E85C1961000000

ARCHIVE LOG COMMAND HISTORY
20:51:19 SEPTEMBER 12, 2012

<table>
<thead>
<tr>
<th>DATE</th>
<th>TIME</th>
<th>RBA</th>
<th>MODE</th>
<th>WAIT</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEP 12, 2012</td>
<td>20:50:57.4</td>
<td>00000000000007FA73A2E</td>
<td>QUIESCE</td>
<td>YES</td>
<td>5 D</td>
</tr>
<tr>
<td>SEP 12, 2012</td>
<td>20:50:49.3</td>
<td>00000000000007FA6F35E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEP 12, 2012</td>
<td>20:50:44.8</td>
<td>00000000000007FA6ACC2</td>
<td>QUIESCE</td>
<td>NO</td>
<td>5 D</td>
</tr>
<tr>
<td>SEP 12, 2012</td>
<td>20:50:39.9</td>
<td>00000000000007FA6B6543</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SYSTEM CCSIDS

---
ASCII SBCS = 1252
ASCII MIXED = 65534
ASCII DBCS = 65534
EBCDIC SBCS = 37
EBCDIC MBCS = 65534
EBCDIC DBCS = 65534
UNICODE SBCS = 367
Example DSNJU004 output for a data sharing member

The sample print log map utility output in the following figure is for a member of a data sharing group.

```
************************************************************************************
  *                                              *
  * LOG MAP OF THE BSDS DATA SET BELONGING TO MEMBER 'DL51 ' OF GROUP 'DSNL5'.  *
  *                                              *
************************************************************************************

DSNJCNVT CONVERSION PROGRAM HAS RUN  DDNAME=GROUP
LOG MAP OF BSDS DATA SET COPY 1, DSN=DSNL5LOG.DL51.BSDS01
LTIME INDICATES LOCAL TIME, ALL OTHER TIMES ARE GMT.
DATA SHARING MODE IS ON
SYSTEM TIMESTAMP - DATE=2013.164  LTIME=13:17:24.34
UTILITY TIMESTAMP - DATE=2013.098  LTIME= 1:06:04.02
VSAM CATALOG NAME=DSNL5SYS
HIGHEST RBA WRITTEN          000000004C45C60F9E9E  2013.164  20:17:47.3
HIGHEST RBA OFFLOADED        000000004C45ACB5CFFF
RBA WHEN CONVERTED TO V4    00000000000000000000
MAX RBA FOR TORBA           00000000000000000000
MIN RBA FOR TORBA           00000000000000000000
STCK TO LRSN DELTA          00000000000000000000

THIS BSDS HAS MEMBER RECORDS FOR THE FOLLOWING MEMBERS:

HOST MEMBER NAME:  DL51
  MEMBER ID:  1
  GROUP NAME:  DSNL5
  BSDS COPY 1 DATA SET NAME:  DSNL5LOG.DL51.BSDS01
  BSDS COPY 2 DATA SET NAME:  DSNL5LOG.DL51.BSDS02
  ENFM START RBA/LRSN:  00C72DC5B25477000000
MEMBER NAME:  DL53
  MEMBER ID:  2
  GROUP NAME:  DSNL5
  BSDS COPY 1 DATA SET NAME:  DSNL5LOG.DL53.BSDS01
  BSDS COPY 2 DATA SET NAME:  DSNL5LOG.DL53.BSDS02
MEMBER NAME:  DL52
  MEMBER ID:  3
  GROUP NAME:  DSNL5
  BSDS COPY 1 DATA SET NAME:  DSNL5LOG.DL52.BSDS01
  BSDS COPY 2 DATA SET NAME:  DSNL5LOG.DL52.BSDS02
MEMBER NAME:  DL54
  MEMBER ID:  4
  GROUP NAME:  DSNL5
  BSDS COPY 1 DATA SET NAME:  DSNL5LOG.DL54.BSDS01
  BSDS COPY 2 DATA SET NAME:  DSNL5LOG.DL54.BSDS02
MEMBER NAME:  DL55
  MEMBER ID:  5
  GROUP NAME:  DSNL5
  BSDS COPY 1 DATA SET NAME:  DSNL5LOG.DL55.BSDS01
  BSDS COPY 2 DATA SET NAME:  DSNL5LOG.DL55.BSDS02
  THIS MEMBER WAS QUIESCED ON 2013.161 AT 14:23:22.9
MEMBER NAME:  DL56
  MEMBER ID:  6
  GROUP NAME:  DSNL5
  BSDS COPY 1 DATA SET NAME:  DSNL5LOG.DL56.BSDS01
  BSDS COPY 2 DATA SET NAME:  DSNL5LOG.DL56.BSDS02
  THIS MEMBER WAS QUIESCED ON 2013.133 AT 18:30:06.9
MEMBER NAME:  DESTROYED
  MEMBER ID:  7
  GROUP NAME:  DSNL5
  BSDS COPY 1 DATA SET NAME:  
  BSDS COPY 2 DATA SET NAME:  
  THIS MEMBER WAS DESTROYED ON 2012.150 AT 17:54:09.5
  THIS MEMBER ID IS AVAILABLE FOR REUSE

**** DISTRIBUTED DATA FACILITY ****
```
COMMUNICATION RECORD

LOCATION=DSNL5 IPNAME=(NULL) PORT=50200 SPORT=50290 RPORT=50201
ALIAS=DSNL5NETT01,DSNL5NETT02,DSNL5NETT03,
DSNL5NETT04,DSNL5NETT05
IPV4=9.30.178.71 IPV6=91E:B247
GRPPV4=9.30.178.60 GRPPV6=91E:B232
LUNAME=STBDL51 PASSWORD=(NULL) GENERICLU=STBDL5G

ACTIVE LOG COPY 1 DATA SETS
<table>
<thead>
<tr>
<th>START RBA/LRNS/TIME</th>
<th>END RBA/LRNS/TIME</th>
<th>DATE/LTIME</th>
<th>DATA SET INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000000004C44A55C00</td>
<td>0000000004C44A55CFF</td>
<td>2011.193</td>
<td>DSN=DSNL5LOG.DLS1.LOGCOPY1.DS03</td>
</tr>
<tr>
<td>00CB108020450A8B860000</td>
<td>00CB1817DEBA05B80000</td>
<td>2011.193</td>
<td>STATUS=REUSABLE</td>
</tr>
<tr>
<td>000000000C44FC39ADFF</td>
<td>000000000C44FC39CFFF</td>
<td>2011.193</td>
<td>DSN=DSNL5LOG.DLS1.LOGCOPY1.DS01</td>
</tr>
<tr>
<td>00CB81C3EE78A060000</td>
<td>00CB81C6DC3E86240000</td>
<td>2011.193</td>
<td>STATUS=TRUNCATED,REUSABLE</td>
</tr>
</tbody>
</table>

ARCHIVE LOG COPY 1 DATA SETS
<table>
<thead>
<tr>
<th>START RBA/LRNS/TIME</th>
<th>END RBA/LRNS/TIME</th>
<th>DATE/LTIME</th>
<th>DATA SET INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000000004C44A45C00</td>
<td>0000000004C44A45CFF</td>
<td>2011.193</td>
<td>DSN=DSNL5LOG.DLS1.LOGCOPY1.DS02</td>
</tr>
<tr>
<td>00CB108020450A8B860000</td>
<td>00CB1817DEBA05B80000</td>
<td>2011.193</td>
<td>STATUS=REUSABLE</td>
</tr>
<tr>
<td>0000000004C44FC39ADFF</td>
<td>0000000004C44FC39CFFF</td>
<td>2011.193</td>
<td>DSN=DSNL5LOG.DLS1.LOGCOPY1.DS02</td>
</tr>
<tr>
<td>00CB108020450A8B860000</td>
<td>00CB1817DEBA05B80000</td>
<td>2011.193</td>
<td>STATUS=REUSABLE</td>
</tr>
</tbody>
</table>

ACTIVE LOG COPY 2 DATA SETS
<table>
<thead>
<tr>
<th>START RBA/LRNS/TIME</th>
<th>END RBA/LRNS/TIME</th>
<th>DATE/LTIME</th>
<th>DATA SET INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000000004C44A55C00</td>
<td>0000000004C44A55CFF</td>
<td>2011.193</td>
<td>DSN=DSNL5LOG.DLS1.LOGCOPY2.DS03</td>
</tr>
<tr>
<td>00CB108020450A8B860000</td>
<td>00CB1817DEBA05B80000</td>
<td>2011.193</td>
<td>STATUS=REUSABLE</td>
</tr>
</tbody>
</table>

934 Utility Guide and Reference
ARCHIVE LOG COPY 2 DATA SETS
NO ARCHIVE DATA SETS DEFINED FOR THIS COPY
CONDITIONAL RESTART CONTROL RECORD
20:17:51 JUNE 13, 2013

**** ACTIVE CRCR RECORD ****
NO CRCR RECORDS ARE ACTIVE

CRCR IDENTIFIER 0003
  USE COUNT 1
  RECORD STATUS
    CRCR NOT ACTIVE
    SUCCESSFUL RESTART
  PROCESSING STATUS
    FORWARD = YES
    BACKOUT = NO
  STARTRBA NOT SPECIFIED
  ENDRBA NOT SPECIFIED
  ENDLRSN NOT SPECIFIED
  ENDTIME NOT SPECIFIED
  EARLIEST REQUESTED RBA 000000000000000000000000
  FIRST LOG RECORD RBA 000000000000000000000000
  ORIGINAL CHECKPOINT RBA 000000000000000000000000
  NEW CHECKPOINT RBA (CHKPTRBA) NOT SPECIFIED
  CRCR CREATED 14:55:09 JANUARY 04, 2012
  END RESTART 14:56:37 JANUARY 04, 2012
  RESTART PROGRESS STARTED ENDED
  CURRENT STATUS REBUILD YES YES
  FORWARD RECOVERY PHASE YES YES
  BACKOUT RECOVERY PHASE YES YES

CRCR IDENTIFIER 0002
  USE COUNT 1
  RECORD STATUS
    CRCR NOT ACTIVE
    SUCCESSFUL RESTART
  PROCESSING STATUS
    COLD START (STARTRBA = ENDRBA)
    FORWARD = NO
    BACKOUT = NO
  STARTRBA 00000000011BE8000000000
  ENDRBA 00000000011BE8000000000
  ENDLRSN NOT SPECIFIED
  ENDTIME NOT SPECIFIED
  EARLIEST REQUESTED RBA 000000000000000000000000
  FIRST LOG RECORD RBA 000000000000000000000000
  ORIGINAL CHECKPOINT RBA 000000000000000000000000
  NEW CHECKPOINT RBA (CHKPTRBA) NOT SPECIFIED
  END RESTART 21:59:35 OCTOBER 16, 2011
  RESTART PROGRESS STARTED ENDED
  CURRENT STATUS REBUILD YES YES
  FORWARD RECOVERY PHASE YES YES
  BACKOUT RECOVERY PHASE YES YES

CRCR IDENTIFIER 0001
  USE COUNT 1
  RECORD STATUS
    CRCR NOT ACTIVE
    SUCCESSFUL RESTART
  PROCESSING STATUS
    FORWARD = NO

Chapter 41. DSNJU004 (print log map)  935
BACKOUT = YES
STARTRBA = NOT SPECIFIED
ENDRBA = NOT SPECIFIED
ENDLSRN = NOT SPECIFIED
ENDTIME = NOT SPECIFIED
EARLIEST REQUESTED RBA = 000000000000000000000000
FIRST LOG RECORD RBA = 000000000000000000000000
ORIGIN CHECKPOINT RBA = 000000000000000000000000

NEW CHECKPOINT RBA (CHKPTRBA) NOT SPECIFIED
CRCCREATED = 17:48:37 MARCH 23, 2010
BEGINRESTART = 17:49:57 MARCH 23, 2010
ENDRESTART = 17:51:28 MARCH 23, 2010
RESTARTPROGRESS = STARTED ENDED

CURRENT STATUS REBUILD = YES YES
FORWARD RECOVERY PHASE = YES YES
BACKOUT RECOVERY PHASE = YES YES

CHECKPOINT QUEUE
20:17:51 JUNE 13, 2013
TIME OF CHECKPOINT = 20:16:43 JUNE 13, 2013
BEGIN CHECKPOINT RBA = 000000004C45C2CE07F2
END CHECKPOINT RBA = 000000004C45C2D9AA15
END CHECKPOINT LRSN = 00CBB1F17D36946E00
TIME OF CHECKPOINT = 20:14:43 JUNE 13, 2013
BEGIN CHECKPOINT RBA = 000000004C45BF6B54F0
END CHECKPOINT RBA = 000000004C45BF781379
END CHECKPOINT LRSN = 00CBB1F10AEE3280400
TIME OF CHECKPOINT = 20:12:43 JUNE 13, 2013
BEGIN CHECKPOINT RBA = 000000004C45B97FCACF
END CHECKPOINT RBA = 000000004C45B98AFE56
END CHECKPOINT LRSN = 00CBB1F09B4E2858AC00

TIME OF CHECKPOINT = 17:02:21 JUNE 13, 2013
BEGIN CHECKPOINT RBA = 000000004C44CDE602F
END CHECKPOINT RBA = 000000004C44CEB8409
END CHECKPOINT LRSN = 00CBB1C60AC5633A60
TIME OF CHECKPOINT = 17:00:21 JUNE 13, 2013
BEGIN CHECKPOINT RBA = 000000004C44C50572CF
END CHECKPOINT RBA = 000000004C44C5FB946
END CHECKPOINT LRSN = 00CBB1C598549E0900
TIME OF CHECKPOINT = 16:58:21 JUNE 13, 2013
BEGIN CHECKPOINT RBA = 000000004C44B8D62C0
END CHECKPOINT RBA = 000000004C44B8E9CC54
END CHECKPOINT LRSN = 00CBB1C525E94ECAE00
TIME OF CHECKPOINT = 16:56:21 JUNE 13, 2013
BEGIN CHECKPOINT RBA = 000000004C44B3623A6
END CHECKPOINT RBA = 000000004C44B37B59
END CHECKPOINT LRSN = 00CBB1C4B37E7D00400
TIME OF CHECKPOINT = 16:54:21 JUNE 13, 2013
BEGIN CHECKPOINT RBA = 000000004C44AC3882B
END CHECKPOINT RBA = 000000004C44ACB704B5
END CHECKPOINT LRSN = 00CBB1C44133B102600

ARCHIVE LOG COMMAND HISTORY
MEMBER DL51
DATA SHARING GROUP DSML5 CONTAINS 7 MEMBERS
20:17:51 JUNE 13, 2013

<table>
<thead>
<tr>
<th>DATE/TIMESTAMP</th>
<th>TIME/STIME</th>
<th>RBA</th>
<th>MODE</th>
<th>WAIT</th>
<th>TIME</th>
<th>SCOPE</th>
<th>CMD</th>
<th>ORIGIN</th>
<th>STATUS</th>
<th>ACTIVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>JUN 10, 2013</td>
<td>13:25:41.9</td>
<td>000000004B74729CE22</td>
<td>QUIESCE</td>
<td>NO</td>
<td>5</td>
<td>D</td>
<td>G</td>
<td>DL51</td>
<td>ORIGIN</td>
<td>5</td>
</tr>
<tr>
<td>JUN 03, 2013</td>
<td>15:12:07.0</td>
<td>000000004A22F3398BE4</td>
<td>QUIESCE</td>
<td>NO</td>
<td>5</td>
<td>D</td>
<td>G</td>
<td>DL51</td>
<td>ORIGIN</td>
<td>5</td>
</tr>
<tr>
<td>MAY 20, 2013</td>
<td>15:54:03.1</td>
<td>000000004B5891AC2858</td>
<td>QUIESCE</td>
<td>NO</td>
<td>5</td>
<td>D</td>
<td>G</td>
<td>DL51</td>
<td>ORIGIN</td>
<td>5</td>
</tr>
<tr>
<td>MAY 13, 2013</td>
<td>16:20:31.7</td>
<td>000000004736159B59</td>
<td>QUIESCE</td>
<td>NO</td>
<td>5</td>
<td>D</td>
<td>G</td>
<td>DL51</td>
<td>ORIGIN</td>
<td>5</td>
</tr>
<tr>
<td>MAY 06, 2013</td>
<td>13:25:22.9</td>
<td>0000000046662D500FF</td>
<td>QUIESCE</td>
<td>NO</td>
<td>5</td>
<td>D</td>
<td>G</td>
<td>DL51</td>
<td>ORIGIN</td>
<td>5</td>
</tr>
<tr>
<td>APR 29, 2013</td>
<td>16:26:52.2</td>
<td>000000004576BE7B44A</td>
<td>QUIESCE</td>
<td>NO</td>
<td>5</td>
<td>D</td>
<td>G</td>
<td>DL51</td>
<td>ORIGIN</td>
<td>5</td>
</tr>
<tr>
<td>APR 22, 2013</td>
<td>14:12:22.4</td>
<td>0000000044250177700</td>
<td>QUIESCE</td>
<td>NO</td>
<td>5</td>
<td>D</td>
<td>G</td>
<td>DL51</td>
<td>ORIGIN</td>
<td>5</td>
</tr>
<tr>
<td>APR 15, 2013</td>
<td>13:25:32.3</td>
<td>00000000435BF4740ECC</td>
<td>QUIESCE</td>
<td>NO</td>
<td>5</td>
<td>D</td>
<td>G</td>
<td>DL51</td>
<td>ORIGIN</td>
<td>5</td>
</tr>
<tr>
<td>APR 08, 2013</td>
<td>13:25:58.9</td>
<td>0000000042589CEAA28A</td>
<td>QUIESCE</td>
<td>NO</td>
<td>5</td>
<td>D</td>
<td>G</td>
<td>DL51</td>
<td>ORIGIN</td>
<td>5</td>
</tr>
<tr>
<td>APR 01, 2013</td>
<td>19:04:21.2</td>
<td>0000000040CD3F02AA1</td>
<td>QUIESCE</td>
<td>NO</td>
<td>5</td>
<td>D</td>
<td>G</td>
<td>DL51</td>
<td>PARTICIPANT</td>
<td>5</td>
</tr>
</tbody>
</table>
Example DSNJU004 output for deactivated and destroyed members

The sample print log map utility output in the following figure is for a deactivated member and a destroyed member of a data sharing group.

Timestamps in the BSDS

The output of the DSNJU004 (print log map) utility includes many timestamps that are recorded in the BSDS. Those timestamps record the date and time of various system events.
Timestamps in the output column LTIME are in local time. All other timestamps are in Coordinated Universal Time (UTC).

The following timestamps are included in the header section of the reports:

**System timestamp**
Reflects the date and time that the BSDS was last updated. The BSDS can be updated by several events:
- Db2 startup.
- The write threshold is reached during log write activities.
  The BSDS might be updated several times a second, or it might not be updated for several seconds, minutes, or even hours. The frequency of updates depends on the number of output buffers that you specified and the system activity rate.
- An error situation where Db2 drops into single-BSDS mode from its normal dual BSDS mode. This action might occur when a request to get, insert, point to, update, or delete a BSDS record is unsuccessful. When this error occurs, Db2 updates the timestamp in the remaining BSDS to force a timestamp mismatch with the disabled BSDS.

**Utility timestamp**
The date and time that the contents of the BSDS were altered by the change log inventory (DSNJU003) utility.

The following timestamps are included in the active and archive log data sets of the reports:

**Active log date**
The date on which the active log data set was originally allocated on the Db2 subsystem.

**Active log time**
The time at which the active log data set was originally allocated on the Db2 subsystem.

**Archive log date**
The date of creation (not allocation) of the archive log data set.

**Archive log time**
The time of creation (not allocation) of the archive log data set.

The following timestamps are included in the conditional restart control record portion of the report that is shown in "DSNJU004 (print log map) output” on page 929:

**Conditional restart control record**
The current time and date. This data is reported for information only and is not kept in the BSDS.

**CRCR created**
The time and date of creation of the CRCR by the CRESTART option in the change log inventory utility.

**Begin restart**
The time and date that the conditional restart was attempted.

**End restart**
The time and date that the conditional restart ended.
STARTRBA (timestamp)
The time at which the control interval was written.

ENDRBA (timestamp)
The time at which the last control interval was written.

Time of checkpoint
The time and date that are associated with the checkpoint record that was used during the conditional restart process.

The following timestamps are included in the checkpoint queue and the DDF communication record sections of the report that is shown in Figure 96 on page 941.

Checkpoint queue
The current time and date. This data is reported for information only and is not kept in the BSDS.

Time of checkpoint
The time and date that the checkpoint was taken.

DDF communication record (heading)
The current time and date. This data is reported for information only, and is not kept in the BSDS.

Active log data set status
The BSDS records the status of an active log data set as one of the status values that are listed in the following table.

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEW</td>
<td>The data set was defined but never used by Db2, or the log is truncated at a point before the data set was created. In either case, the data set starting and ending RBA values are reset to zero.</td>
</tr>
<tr>
<td>REUSABLE</td>
<td>Either the data set is new and has no records, or the data set was offloaded. In the print log map output, the start RBA value for the last REUSABLE data set is equal to the start RBA value of the last archive log data set.</td>
</tr>
<tr>
<td>NOT REUSABLE</td>
<td>The data set contains records that have not been offloaded.</td>
</tr>
<tr>
<td>STOPPED</td>
<td>The offload processor encountered an error while reading a record, and that record could not be obtained from the other copy of the active log. Alternatively, an error occurred during truncation of the data set following a write I/O error.</td>
</tr>
<tr>
<td>TRUNCATED</td>
<td>One of these conditions exists:</td>
</tr>
<tr>
<td></td>
<td>• An I/O error occurred, and Db2 stopped writing to this data set. The active log data set is offloaded, beginning with the starting RBA and continuing to the last valid record segment in the truncated active log data set. (The RBA of the last valid record segment is less than the ending RBA of the active log data set.) Logging is switched to the next available active log data set and continues uninterrupted.</td>
</tr>
<tr>
<td></td>
<td>• The log was truncated by a conditional restart at a point within the data set RBA range.</td>
</tr>
<tr>
<td></td>
<td>• The Db2 ARCHIVE LOG command was issued while this data set was the current active log data set.</td>
</tr>
</tbody>
</table>

The status value for each active log data set is displayed in the print log map utility output, as shown in the following example output:
Archive log command history

The print log map utility output also displays the archive log command history, as shown in the following figure.

```
<table>
<thead>
<tr>
<th>DATE/TIME</th>
<th>RBA/LRSN/TIME</th>
<th>END RBA/LRSN/TIME</th>
<th>DATE/LTIME</th>
<th>DATA SET INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>2012.219</td>
<td>19:35:03.9</td>
<td>2012.219 19:35:09.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000000000027DA5000</td>
<td>000000000000027DA5FF</td>
<td>2011.122 DSN=DSNC000.DB2A.LOGCOPY1.DS02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00C9FAE4278FF00000000</td>
<td>00C9FAE4278FF00000000</td>
<td>16:49 STATUS=REUSABLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2012.000</td>
<td>0:00:00:0.0</td>
<td>0000.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000000000000283660000</td>
<td>000000000000028366FFF</td>
<td>2011.122 DSN=DSNC000.DB2A.LOGCOPY1.DS01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00CA28653CD94000000</td>
<td>00CA28653CD94000000</td>
<td>16:49 STATUS=REUSABLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2012.256</td>
<td>00:12:15.8</td>
<td>000000000000029F15FFF</td>
<td>2011.122 DSN=DSNC000.DB2A.LOGCOPY1.DS01</td>
<td></td>
</tr>
<tr>
<td>00C9FAE4278FF00000000</td>
<td>00C9FAE4278FF00000000</td>
<td>16:49 STATUS=REUSABLE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**Figure 94. Portion of print log map utility output that shows active log data set status**

Reading conditional restart control records

The print log map utility also lists information about each conditional restart control record and each checkpoint. A sample description of a checkpoint record in the queue is shown in the following figure.

```
<table>
<thead>
<tr>
<th>DATE/LTIME</th>
<th>TIME/LTIME</th>
<th>DATA SET INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>20:50:57.4</td>
<td>0000000000007FA73AE</td>
<td>QUIESCE YES 5 D</td>
</tr>
<tr>
<td>20:50:49.3</td>
<td>0000000000007FA6F3E</td>
<td>QUIESCE NO 5 D</td>
</tr>
<tr>
<td>20:50:44.8</td>
<td>0000000000007FA6ACC2</td>
<td>QUIESCE NO 5 D</td>
</tr>
<tr>
<td>20:50:39.9</td>
<td>0000000000007FA66543</td>
<td>QUIESCE NO 5 D</td>
</tr>
</tbody>
</table>
```

**Figure 95. Portion of print log map utility output that shows archive log command history**
CHECKPOINT QUEUE
19:31:37 SEPTEMBER 12, 2012
BEGIN CHECKPOINT RBA  00000000000028C12842
END CHECKPOINT RBA  00000000000028C16B7A
END CHECKPOINT LRSN  00CA296B55DA8E858000
BEGIN CHECKPOINT RBA  000000000000028C0C280
END CHECKPOINT RBA  00000000000028C1057A
END CHECKPOINT LRSN  00CA296B4FE13EAC000
TIME OF CHECKPOINT  19:31:06 SEPTEMBER 12, 2012
BEGIN CHECKPOINT RBA  000000000000028C059FE
END CHECKPOINT RBA  00000000000028C09C7A
END CHECKPOINT LRSN  00CA296B4275C34C4000
...
TIME OF CHECKPOINT  19:34:52 AUGUST 06, 2012
BEGIN CHECKPOINT RBA  0000000000000243F3D36
END CHECKPOINT RBA  0000000000000243F8C26
END CHECKPOINT LRSN  00C9FAE41852F200000

Figure 96. Sample print log map description of checkpoints

A sample description of a conditional restart control record is shown in the following figure.
System-level backup information

The print log map utility also displays information about system-level backup copies that are created by the BACKUP SYSTEM utility. An example of system-level backup information is shown in the following figure.

![Sample print log map description of a CRCR](image-url)
When a log copy pool is restored, accurate date and time values are not displayed for the system-level backup that is used to restore the copy pool. Instead, the DATA/LOG DATE value is displayed as 0000/00/00, and the COMPLETE LTIME value is displayed as 00:00:00. The reason is that this information is not recorded in the BSDS until after the system-level backup is complete. Therefore, this information is not available at the time that the backup copy of the BSDS is made.

The following figure shows an example of an incremental system-level backup. TYPE applies to only the database copy pool history entries.

Figure 98. System-level backup information

![Backup System Utility History](backup_system_utility_history.png)

Figure 99. System-level backup information as an incremental copy

Related concepts:
- Management of the bootstrap data set (Db2 Administration Guide)
- Conditional restart (Db2 Administration Guide)

Related tasks:
- Deleting data sharing members (Db2 Data Sharing Planning and Administration)
Restoring deactivated data sharing members (Db2 Data Sharing Planning and Administration)

Related reference:
Chapter 8, “BACKUP SYSTEM,” on page 51
Chapter 42. DSNJU008 (print CDDS)

The DSNJU008 (print CDDS) stand-alone utility prints the compression dictionary data set (CDDS).

Environment

The DSNJU008 program runs as a batch job.

This utility can be executed when Db2 is running or when it is not running. DSNJU008 can be executed on the source Db2 data sharing group or the proxy Db2 data sharing group in an implementation of the GDPS Continuous Availability with zero data loss solution.

Authorization required

The user ID of the DSNJU008 job must have requisite RACF authorization.

Required data sets

DSNJU008 recognizes DD statements with the following DD names:

- **SYSUT1**
  Specifies and allocates the compression dictionary data set (CDDS). This statement is required. If the CDDS must be shared with a concurrently executing Db2 online subsystem, use DISP=SHR in the DD statement.
  
  If the CDDS must be shared on the proxy Db2 data sharing group, also specify ROACCESS=(ALLOW,TRKLOCK) in the DD statement.

- **SYSPRINT**
  Specifies a data set or print spool class for print output. This statement is required. The logical record length (LRECL) is 132.

- **SYSIN**
  Contains the control statement for the DSNJU008 utility.

Running the DSNJU008 utility

Use the following EXEC statement to execute this utility:

```plaintext
// EXEC PGM=DSNJU008
```

Related tasks:

- Reading complete log data for the GDPS Continuous Availability with zero data loss solution (Db2 Administration Guide)

Related reference:

- DD statement (MVS JCL Reference)

Syntax and options of the DSNJU008 control statement

The DSNJU008 control statement that is specified in the SYSIN data set allows you to specify the records that are printed, and how the records are formatted.
Option descriptions

The following keywords can be used in a control statement in the SYSIN data set:

**DBID(hex-constant)**

Specifies a hexadecimal database identifier (DBID). DSNJU008 prints only the records in the compression dictionary data set (CDDS) that are associated with that DBID. Specify this keyword no more than once in the job.

`hex-constant` is a hexadecimal value that consists of one to four characters. Leading zeros are not required.

You can find the DBID in the following ways:
- The DBID is displayed in many Db2 messages.
- You can find the DBID in the Db2 catalog for a specific object (for example, in the column that is named DBID of the SYSIBM.SYSTABLESPACE catalog table).

When you select a DBID from a catalog table, the value is displayed in decimal format. Use the SQL HEX function in a SELECT statement to convert a DBID to hexadecimal format. The following SQL statements show this use of the HEX function:

```sql
SELECT NAME, DBNAME, HEX(DBID), HEX(PSID)
FROM SYSIBM.SYSTABLESPACE
WHERE NAME = 'table-space-name'
```

- You can use the DSN1PRNT utility to format the data sets for tables or indexes, and find the DBID in the first two bytes of HPGOBID.

**PSID(hex-constant)**

Specifies a hexadecimal page set identifier (PSID) of a table space. If you specify the PSID keyword, you must also specify the DBID keyword.

DSNJU008 prints only the expansion dictionary records that are associated with the DBID and the PSID. Specify this keyword no more than once in the job.

`hex-constant` is a hexadecimal value that consists of one to four characters. Leading zeros are not required.

You can find the PSID in the following ways:
- Whenever Db2 changes the data, the log record that describes the change identifies the database by the DBID and the table space by the PSID. You can find the PSID column in the SYSIBM.SYSTABLESPACE catalog table.
When you select a PSID from a catalog table, the value is displayed in decimal format. Use the SQL HEX function in a SELECT statement to convert the PSID to hexadecimal format.

- You can use the DSN1PRNT utility to format the data sets for tables, and find the PSID in the last two bytes of HPGOBID.

**PART(hex-constant)**

Specifies a hexadecimal partition number identifier, which is associated with a database identifier (DBID) and a data page set identifier (PSID). DSNJU008 prints only the records that are associated with the specified partition number.

*hex-constant* is a hexadecimal value that consists of one to four characters. Leading zeros are not required.

Specify the PART keyword no more than once in the job. When you specify the PART keyword, you must also specify the DBID and the PSID keywords that correspond to the partitions that are identified by the PART keyword. If the PART keyword is not specified, records for all partitions that are associated with the DBID and PSID are printed.

**VER(integer-constant)**

Specifies the version of the expansion dictionary that is to be printed. This value must be 1, 2, or 3.

Up to three versions of the expansion dictionary are kept in the CDDS for an object. Version 1 is the most recent expansion dictionary. When you specify the VER keyword, DSNJU008 prints only the records for the specified expansion dictionary version.

Specify the VER keyword no more than once in the job. If you do not specify the VER keyword, DSNJU008 prints the records for all expansion dictionary versions.

**DUMP(YES|NO)**

Specifies whether hexadecimal output for each expansion dictionary record is printed.

**YES**

Specifies that hexadecimal output of expansion dictionary records is printed.

**NO**

Specifies that hexadecimal output of expansion dictionary records is not printed. NO is the default.

Specify the DUMP keyword no more than once in the job.

**SYSTEM(YES|NO|ONLY)**

Specifies whether hexadecimal output of system records in the CDDS is printed. System records are used for managing the CDDS. The output of the system records is for use by IBM Support.

**YES**

Specifies that hexadecimal output of system records in the CDDS is printed.

**NO**

Specifies that hexadecimal output of system records is not printed. NO is the default.

**ONLY**

Specifies that hexadecimal output of only the system records in the CDDS is printed. Expansion dictionary records are not printed.
When SYSTEM(YES) or SYSTEM(ONLY) is specified, hexadecimal output of system records is printed, regardless of the DUMP setting.

Specify the SYSTEM keyword no more than once in the job.

**LIMIT(integer-constant)**

Specifies the maximum number of lines of output that the DSNJU008 utility prints.

*integer-constant* is an integer value of one to 10 characters. The maximum value is 4294967295, which is also the default.

DSNJU008 always prints complete expansion dictionary records, which can cause the number of lines that are printed to exceed *integer-constant*. If part of a record is not printed when *integer-constant* is reached, DSNJU008 prints the rest of that record.

Specify the LIMIT keyword no more than once in the job.

### DSNJU008 examples

Use the DSNJU008 examples to learn how to write your own DSNJU008 jobs.

**Example DSNJU008 input and output**

The following statement specifies that DSNJU008 is to print information from CDDS data set TEST.CDDS about the most recent version of the expansion dictionary for database ID (DBID) x’011D’ and page set ID (PSID) x’0002’.

```
//PLM8
EXEC PGM=DSNJU008
//SYSPRINT DD SYSOUT=*  
//SYSUT1 DD DSN=TEST.CDDS,DISP=SHR
//SYSIN DD *  
   DBID(011D) PSID(0002) VER(1)
```

The output from this job looks similar to the following output.

```
COMPRESSION DICTIONARY DATA SET (CDDS)
CDDS STATE       ACTIVE
NUMBER OF MEMBERS 002
HIGH MEMBER ID    002
NUMBER OF ACTIVE MEMBERS 002
CDDS LAST UPDATE 2018.07.02 20:27:46.5
CDDS STATE LAST UPDATE 2018.07.02 19:58:32.2
DICTIONARY LAST UPDATE 2018.07.02 20:06:01.5
STCK TO LRSN DELTA 0

001 DB2A ACTIVE 2018.07.02 20:26:35.1 2018.07.02 20:26:23.0 DSNCO00.0B2A.BSDS01
002 DB2B ACTIVE 2018.07.02 20:27:46.5 2018.07.02 20:27:32.5 DSNCO00.0B2B.BSDS01

DE0101D00D2B0008 0110 0002 0008 01 DBB .TSS 00D405D06A6D4B981600 2018.07.02 15:16:25 00D405D06A6D4B981600 2018.07.02 15:16:25

NUMBER OF RECORDS PRINTED 1
DICTIONARY RECORDS PRINTED 1
SYSTEM RECORDS PRINTED 0
DSNJ225I DSNJU008 PRINT OPERATION COMPLETED SUCCESSFULLY
DSNJ200I DSNJU008 PRINT CDDS UTILITY PROCESSING COMPLETED SUCCESSFULLY
```

---
The DSN1COMP stand-alone utility estimates space savings that are to be achieved by Db2 data compression in table spaces and indexes.

You can run this utility on the following types of data sets that contain uncompressed data:

- Db2 full image copy data sets
- VSAM data sets that contain Db2 table spaces
- Sequential data sets that contain Db2 table spaces (for example, DSN1COPY output)

**Restrictions:**

You cannot run DSN1COMP on concurrent copies.

DSN1COMP does not estimate savings for data sets that contain LOB table spaces.

If you run DSN1COMP on a table space in which the data is the same for all rows, message DSN1941I is issued. In this case, DSN1COMP does not compute any statistics.

**Environment**

Run DSN1COMP as a z/OS job.

You can run DSN1COMP even when the Db2 subsystem is not operational. Before you use DSN1COMP when the Db2 subsystem is operational, issue the Db2 STOP DATABASE command. Issuing the STOP DATABASE command ensures that Db2 has not allocated the Db2 data sets.

Do not run DSN1COMP on table spaces in DSNDB01, DSNDB06, or DSNDB07.

**Authorization required**

DSN1COMP does not require authorization. However, if any of the data sets is RACF-protected, the authorization ID of the job must have RACF authority.

**Required data sets**

DSN1COMP uses the following data definition (DD) statements:

**SYSPRINT**

Defines the data set that contains output messages from DSN1COMP and all hexadecimal dump output.

**SYSUT1**

Defines the input data set, which can be a sequential data set or a VSAM data set.

Specify the disposition for this data set as OLD (DISP=OLD) to ensure that it is not in use by Db2. Specify the disposition for this data set as SHR (DISP=SHR) only in circumstances where the Db2 STOP DATABASE command does not work.
The requested operation takes place only for the specified data set. In the following situations, you must specify the correct data set.

- The input data set belongs to a nonpartitioned table space.
- The index space is larger than 2 GB.
- The table space or index space is a partitioned space.

If you are running the online REORG utility with FASTSWITCH behavior, verify the data set name before running the DSN1COMP utility. The fifth-level qualifier in the data set name alternates between 'I0001' and 'J0001' when using FASTSWITCH. If the table space has cloning, the fifth-level qualifier can be 'n0002.' You cannot specify FASTSWITCH YES if the table space has cloning; however, a FASTSWITCH YES REORG might have been done before the clone was created so you might still have a mixture of 'I' and 'J' data sets. Specify the correct fifth-level qualifier in the data set name to successfully execute the DSN1COMP utility. To determine the correct fifth-level qualifier, query the IPREFIX column of SYSIBM.SYSTABLEPART for each data partition or the IPREFIX column of SYSIBM.SYSINDEXPART for each index partition. If the object is not partitioned, use zero as the value for the PARTITION column in your query.

**DSN1DICT**

DSN1DICT is required only if you specify the EXTNDICT parameter, to create an external copy of the compression dictionary that DSN1COMP produces.

DSN1DICT defines the output data set to which the external copy of the compression dictionary is written. This data set must:

- Be a sequential data set or a member of a partitioned data set
- Have fixed record format with a record length of 80

The data set or data set member that is produced is an object module that can be link-edited into a program.

**Recommendation**

Before using DSN1COMP, be sure that you know the page size and data set size (DSSIZE) for the table space. Use the following query on the Db2 catalog to get the information you need, in this example for table 'DEPT':

```sql
SELECT T.CREATOR, T.NAME, S.NAME AS TABLESPACE, S.PARTITIONS, S.PGSIZE,
CASE S.DSSIZE
  WHEN 0 THEN
    CASE WHEN S.TYPE = 'G' THEN 4194304
          WHEN S.TYPE = 'O' THEN 4194304
          WHEN S.TYPE = 'P' THEN 4194304
          WHEN S.TYPE = 'R' THEN 4194304
          ELSE 2097152
      END
  WHEN S.PARTITIONS > 254 THEN
    CASE WHEN S.PGSIZE = 4 THEN 4194304
          WHEN S.PGSIZE = 8 THEN 8388608
          WHEN S.PGSIZE = 16 THEN 16777216
          WHEN S.PGSIZE = 32 THEN 33554432
          ELSE NULL
      END
    WHEN S.PARTITIONS > 64 THEN 4194304
    WHEN S.PARTITIONS > 32 THEN 1048576
    WHEN S.PARTITIONS > 16 THEN 2097152
    WHEN S.PARTITIONS > 0 THEN 4194304
    ELSE 2097152
  END
END
```
Syntax and options of the DSN1COMP control statement

The DSN1COMP utility control statement, with its multiple options, defines the function that the utility job performs.

**DSN1COMP syntax diagram**

For table spaces:

```
│ DSN1COMP  │ 32K  │ PAGESIZE  │ DSSIZE  │ NUMPARTS(integer)  │
│        │      │     (integer) │      │                   │
│        │      │      4K      │      │                   │
│        │      │      8K      │      │                   │
│        │      │      16K     │      │                   │
│        │      │      32K     │      │                   │
│        │      │            │      │                   │
│        │      │            │      │ FREEPAGE(integer)  │
│        │      │            │      │ PCTFREE(integer)   │
│        │      │            │      │ FULLCOPY           │
│        │      │            │      │ REORG              │
│        │      │            │      │ ROWLIMIT(integer)  │
│        │      │            │      │                   │
│        │      │            │      │ MAXROWS(integer)   │
│        │      │            │      │ EXTNDCIT(dictionary-name)  │
│        │      │            │      │ LOB                │
```

For indexes:

```
│ DSN1COMP  │ LEAFLIM(integer)  │
```

**Option descriptions**

To run DSN1COMP, specify one or more of the following parameters on the EXEC statement to run DSN1COMP. If you specify more than one parameter, separate each parameter by a comma. You can specify parameters in any order.

**32K**

Specifies that the input data set, SYSUT1, has a 32-KB page size. If you specify this option and the SYSUT1 data set does not have a 32-KB page size, DSN1COMP might produce unpredictable results.

The recommended option for performance is `PAGESIZE(32K)`.
PAGESIZE
Specifies the page size of the input data set that is defined by SYSUT1.
Available page size values are 4K, 8K, 16K, or 32K. If you specify an incorrect
page size, DSN1COMP might produce unpredictable results.
If you omit PAGESIZE, DSN1COMP tries to determine the page size from the
input data set. Db2 issues an error message if DSN1COMP cannot determine
the input page size. This might happen if the header page is not in the input
data set, or if the page size field in the header page contains an invalid page
size.
If information on this value is available in the input data set header page, the
header page information is the default.

DSSIZE(integer G)
Specifies the data set size, in gigabytes, for the input data set. If you omit
DSSIZE, Db2 obtains the data set size from the data set header page.
If you specify DSSIZE, integer must match the DSSIZE value that was specified
when the table space was defined.

LARGE
Specifies that the input data set is a table space that was defined with the
LARGE option. If you specify LARGE, Db2 assumes that the data set has a
4-GB boundary.
The recommended method of specifying a table space defined with LARGE is
DSSIZE(4G).
If you omit the LARGE or DSSIZE(4G) option when it is needed, or if you
specify LARGE for a table space that was not defined with the LARGE option,
the results from DSN1COMP are unpredictable.
If information on this value is available in the input data set header page, the
header page information is the default.

NUMPARTS(integer)
Specifies the number of partitions that are associated with the input data set.
Valid specifications range 1 - 4096. If you omit NUMPARTS or specify it as 0,
DSN1COMP assumes that your input file is not partitioned. If you specify a
number greater than 64, DSN1COMP assumes that the data set is for a
partitioned table space that was defined with the LARGE option, even if the
LARGE keyword is not specified.
DSN1COMP cannot always validate the NUMPARTS parameter. If you specify
it incorrectly, DSN1COMP might produce unpredictable results.

DSN1COMP terminates and issues message DSN1946I when it encounters an
image copy that contains multiple partitions; a compression report is issued for
the first partition.
This parameter is not used if the target table space is a universal table space.
DSSIZE is used instead.
This parameter is deprecated.

FREEPAGE(integer)
Specifies how often to leave a page of free space when calculating the
percentage of saved pages. You must specify an integer in the range 0 to 255. If
you specify 0, no pages are included as free space when DSN1COMP reports
the percentage of pages saved. Otherwise, one free page is included after every
n pages, where n is the specified integer.
The default value is 0.

Specify the same value that you specify for the FREEPAGE option of the SQL statement CREATE TABLESPACE or ALTER TABLESPACE.

**PCTFREE(integer)**

Indicates what percentage of each page to leave as free space when calculating the percentage of pages saved. You must specify an integer in the range 0 to 99. When calculating the savings, DSN1COMP allows for at least \( n \) percent of free space for each page, where \( n \) is the specified integer.

The default value is 5.

Specify the same value that you specify for the PCTFREE option of the SQL statement CREATE TABLESPACE or ALTER TABLESPACE.

**FULLCOPY**

Specifies that a Db2 full image copy (not a DFSMSdss concurrent copy) of your data is to be used as input. Omitting this parameter when the input is a full image copy can cause error messages or unpredictable results. If this data is partitioned, also specify the NUMPARTS parameter to identify the number of partitions.

**REORG**

Provides an estimate of compression savings that are comparable to the savings that the REORG utility would achieve. If this keyword is not specified, the results are similar to the compression savings that the LOAD utility would achieve.

**ROWLIMIT(integer)**

Specifies the maximum number of rows to evaluate in order to provide the compression estimate. This option prevents DSN1COMP from examining every row in the input data set. Valid specifications range from 1 to 99000000.

Use this option to limit the elapsed time and processor time that DSN1COMP requires. An analysis of the first 5 to 10 MB of a table space provides a fairly representative sample of the table space for estimating compression savings. Therefore, specify a ROWLIMIT value that restricts DSN1COMP to the first 5 to 10 MB of the table space. For example, if the row length of the table space is 200 bytes, specifying ROWLIMIT(50000) causes DSN1COMP to analyze approximately 10 MB of the table space.

**MAXROWS(integer)**

Specifies the maximum number of rows that DSN1COMP is to consider when calculating the percentage of pages saved. You must specify an integer in the range 1 to 255.

The default value is 255.

Specify the same value that you specify for the MAXROWS option of the SQL statement CREATE TABLESPACE or ALTER TABLESPACE.

**EXTNDICT(dictionary-name)**

Specifies the name of an external copy of the compression dictionary that DSN1COMP produces. `dictionary-name` must:

- Be eight bytes
- Contain only uppercase alphanumeric characters
- Begin with an alphabetic character

The external copy of the compression dictionary is primarily for use by the IBM Data Encryption for IMS and Db2 tool.
When EXTNDICT is specified, a DSN1DICT DD statement must be included in the JCL for running DSN1COMP.

**LOB**

Specifies that the input data set is a LOB table space. When LOB is specified, DSN1COMP will estimate the space savings and compression ratio for a LOB table space using the zEnterprise data compression (zEDC) hardware.

If the LOB table space is not compressed, DSN1COMP will calculate the compression savings that would occur if the table space is defined using the COMPRESS YES option. DSN1COMP will collect data up to the average LOB size (or 1 MB as the maximum size) and pass the collected data to the zEDC card (if zEDC hardware is installed), which will return the compressed information and statistics for the DSN1COMP report.

If the LOB table space is already compressed (the table space is defined with the COMPRESS YES option), DSN1COMP will collect statistics of the individual LOB metadata from the LOB map page without collecting and compressing data using the zEDC card again.

**LEAFLIM**(integer)

Specifies how many index leaf pages should be evaluated to determine the compression estimate. This option prevents DSN1COMP from processing all index leaf pages in the input data set. Valid specifications range from 1 to 99000000.

If the LEAFLIM parameter is not specified, the entire index will be scanned and all leaf pages will be examined.

In a compressed index, only leaf pages are compressed. All other page types remain uncompressed.

Related reference:

[IBM InfoSphere Guardium Data Encryption for IMS and Db2 Databases](#)

---

**Before running DSN1COMP**

Certain activities might be required before you run the DSN1COMP utility, depending on your situation.

If you run DSN1COMP on a segmented table space, you must first query the SYSTABLEPART catalog table to determine the current instance qualifier, which is stored in the IPREFIX column. You can then use the current instance qualifier to code the data set name in the JCL. The following sample shows an example of such a query.

```
SELECT DBNAME, TSNAME, PARTITION, IPREFIX
FROM SYSIBM.SYSTABLEPART
WHERE DBNAME = 'DBMC0731' AND TSNAME = 'TPMC0731'
ORDER BY TSNAME, PARTITION;
```

The preceding query produces the following result:
How to estimate compression savings achieved with option REORG

If you run DSN1COMP with the REORG option on small data sets, the resulting estimates might vary greatly from the estimates that are produced without the default REORG option. Alternatively, if you run DSN1COMP and specify a small number (n) for ROWLIMIT, the estimates might vary greatly from the estimates that are produced without REORG.

DSN1COMP does not try to convert data to the latest version before it compresses rows and derives a savings estimate.

Without the REORG option, DSN1COMP uses the first n rows to fill the compression dictionary. DSN1COMP processes the remaining rows to provide the compression estimate. If the number of rows that are used to build the dictionary is a significant percentage of the data set rows, little savings result. With the REORG option, DSN1COMP processes all the rows, including those that are used to build the dictionary, which results in greater compression.

The DSN1COMP utility determines possible saving estimates at the data set level for a unique partition only. Therefore, if DSN1COMP is run against an image copy data set that contains several partitions or against a single partition of partition-by-growth table spaces (PBGs), the results will be different from what the REORG utility would produce.

Free space in compression calculations on table space

The DSN1COMP utility makes compression estimates, which take into account the PCTFREE and FREEPAGE options.

If you use different PCTFREE or FREEPAGE values than those that were created with the input table space, you get a different value for noncmpages. DSN1COMP reports this value in message DSN1940I, as shown in the example output in the following figure.

```
+-------------------------------------------+ | DBNAME | TSNAME | PARTITION | IPREFIX |
+-------------------------------------------+ 1   | DBMC0731 | TPMC0731  | 1 | J   |
 2   | DBMC0731 | TPMC0731  | 2 | J   |
 3   | DBMC0731 | TPMC0731  | 3 | J   |
 4   | DBMC0731 | TPMC0731  | 4 | J   |
 5   | DBMC0731 | TPMC0731  | 5 | J   |
+-------------------------------------------+
```
NO ROWLIMIT WAS REQUESTED
ESTIMATE BASED ON DB2 LOAD METHOD

DSN1940I DSNICOMP COMPRESSION REPORT
1,289 KB WITHOUT COMPRESSION
717 KB WITH COMPRESSION
44 PERCENT OF THE BYTES WOULD BE SAVED

176 ROWS SCANNED TO BUILD DICTIONARY
20,000 ROWS SCANNED TO PROVIDE COMPRESSION Estimate
512 DICTIONARY ENTRIES

1 DICTIONARY PAGES REQUIRED
147 PAGES REQUIRED WITHOUT COMPRESSION
148 PAGES REQUIRED WITH COMPRESSION
0 PERCENT OF THE DB2 DATA PAGES WOULD BE SAVED

*** DETAIL REPORT OF FREQUENCIES AND AVERAGES ***

1 CHILD CHARACTER WAS COMPARED  566,764 TIMES
2 CHILD CHARACTERS WERE COMPARED  182,026 TIMES
3 CHILD CHARACTERS WERE COMPARED  10,300 TIMES
5 CHILD CHARACTERS WERE COMPARED  1,129 TIMES
TOTAL ALPHABET NODE COMPARISONS  528,139 TIMES

967,361 CHILD COMPARISONS IN THE SIBLING LISTS
760,219 SEARCHES IN THE SIBLING LISTS
1.2 AVERAGE NUMBER OF COMPARISONS PER SEARCH
60 BYTES FOR AVERAGE UNCOMPRESSED ROW LENGTH
39 BYTES FOR AVERAGE COMPRESSED ROW LENGTH

263 IS THE DATABASE ID (DBID)
2 IS THE PAGESET ID (PSID)

Sample DSN1COMP control statements

Use the sample control statements as models for developing your own DSN1COMP control statements.

Example 1: Estimating space savings from data compression for a full image copy

The following statement specifies that the DSN1COMP utility is to report the estimated space savings that are to be achieved by compressing the full image copy that is identified by the SYSUT1 DD statement. In this statement, the DSN value specifies the data set name of the image copy that is to be used as input. Because the input is a full image copy, the FULLCOPY parameter must be specified.

//jobname JOB acct information
//COMPEST EXEC PGM=DSN1COMP,PARM='FULLCOPY'
//STEPLIB DD DSN=prefix.SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSABEND DD SYSOUT=A
//SYSUT1 DD DSN=COPY001F.IFDY01,DISP=SHR

Example 2: Providing intended free space when estimating space savings

In the following sample statement, STEP1 specifies that DSN1COMP is to report the estimated space savings that are to be achieved by compressing the data in the data set that is identified by the SYSUT1 DD statement, DSNC810.DSNDBD.DB254SP4.TS254SP4.I0001.A00. When calculating these
estimates, DSN1COMP considers the values passed by the PCTFREE and FREEPAGE options. The PCTFREE value indicates that 20% of each page is to be left as free space. The FREEPAGE value indicates that every fifth page is to be left as free space. This value must be the same value that you specified for the FREEPAGE option of the SQL statement CREATE TABLESPACE or ALTER TABLESPACE.

STEP2 specifies that DSN1COMP is to report the estimated space savings that are to be achieved by compressing the data in the data set that is identified by the SYSUT1 DD statement, DSNC810.DSNDBD.DB254SP4.TS254SP4.I0001.A0001. When providing the compression estimate, DSN1COMP is to evaluate no more than 20,000 rows, as indicated by the ROWLIMIT option. Specifying the maximum number of rows to evaluate limits the elapsed time and processor time that DSN1COMP requires.

Example 3: Estimating space savings that are comparable to what the REORG utility would achieve.

The following statement specifies that DSN1COMP is to report the estimated space savings that are to be achieved by compressing the data in the data set that is identified by the SYSUT1 DD statement, DSNCAT.DSNDBD.DB2J0201.TP201.I0001.A254. This input data set is a table space that was defined with the LARGE option and has 254 partitions, as indicated by the DSN1COMP options LARGE and NUMPARTS.

The REORG option indicates that DSN1COMP is to provide an estimate of compression savings that is comparable to the savings that the REORG utility would achieve, rather than what the LOAD utility would achieve.

When calculating these estimates, DSN1COMP considers the values passed by the PCTFREE and FREEPAGE options. The PCTFREE value indicates that 30% of each page is to be left as free space. The FREEPAGE value indicates that every thirtieth page is to be left as free space. This value must be the same value that you specified for the FREEPAGE option of the SQL statement CREATE TABLESPACE or ALTER TABLESPACE. DSN1COMP is to evaluate no more than 20,000 rows, as indicated by the ROWLIMIT option.
Example 4: Building an Object Module from the DSN1COMP generated dictionary.

In the sample statement, BUILD specifies that DSN1COMP is to externalize the compression dictionary that it generated. This behavior is indicated by the EXTNDICT option, which requires that a DSN1DICT DD statement be provided. DSN1DICT identifies the output data set to which the generated object module is written and stored for additional processing.

```
//BUILD EXEC PGM=DSN1COMP,
//      PARM='DSSIZE(4G),EXTNDICT(dictname),'
//SYSPRINT DD SYSOUT=A
//SYSTUT DD DSN=DSNDSNDBD.DBJT0201.TPIJ0201.10000.A254,DISP=SHR
//      DISP=SHR
//DSN1DICT DD DSN=&&OBJ,
//      DISP=(,PASS),
//      UNIT=SYSALLDA,SPACE=(TRK,(8,4)),
//      DCB=(LRECL=80,BLKSIZE=4000,RECFM=FB)
```

Figure 103. Example DSN1COMP statement with the EXTNDICT option.

**DSN1COMP output**

The output of the DSN1COMP utility lists a variety of information.

**Message DSN1941**

If you receive this message, use a data set with more rows as input, or specify a larger ROWLIMIT.

**Sample DSN1COMP report**

The following figure shows a sample of the output that DSN1COMP generates.
DSN1940I DSN1COMP COMPRESSION REPORT
301 KB WITHOUT COMPRESSION
224 KB WITH COMPRESSION
25 PERCENT OF THE BYTES WOULD BE SAVED

1,975 ROWS SCANNED TO BUILD DICTIONARY
4,665 ROWS SCANNED TO PROVIDE COMPRESSION ESTIMATE
4,096 DICTIONARY ENTRIES

81 BYTES FOR AVERAGE UNCOMPRESSED ROW LENGTH
52 BYTES FOR AVERAGE COMPRESSED ROW LENGTH

16 DICTIONARY PAGES REQUIRED
110 PAGES REQUIRED WITHOUT COMPRESSION
99 PAGES REQUIRED WITH COMPRESSION
10 PERCENT OF THE DB2 DATA PAGES WOULD BE SAVED

Figure 104. Sample DSN1COMP report

Example of DSN1COMP on an index

The following figure shows an example of the output that DSN1COMP on an index generates.

Figure 105. Sample DSN1COMP output

DSN1999I START OF DSN1COMP FOR JOB DSN1COMP STEP2
DSN1998I INPUT DSNAME = TESTCAT.DSNDBD.DB1COMPR.I1.I0001.A001 , VSAM

DSN1944I DSN1COMP INPUT PARAMETERS
PROCESSING PARMS FOR INDEX DATASET:
NO LEAFLIM WAS REQUESTED

DSN1940I DSN1COMP COMPRESSION REPORT
38 Index Leaf Pages Processed
3,000 Keys Processed
3,000 Rids Processed
401 KB of Key Data Processed
106 KB of Compressed Keys Produced

EVALUATION OF COMPRESSION WITH DIFFERENT INDEX PAGE SIZES:

----------------------------------------------
8 K Page Buffer Size yields a
51 % Reduction in Index Leaf Page Space
The Resulting Index would have approximately
49 % of the original index's Leaf Page Space
No Bufferpool Space would be unused
----------------------------------------------

----------------------------------------------
16 K Page Buffer Size yields a
74 % Reduction in Index Leaf Page Space
The Resulting Index would have approximately
26 % of the original index's Leaf Page Space
3 % of Bufferpool Space would be unused to
ensure keys fit into compressed buffers
----------------------------------------------

DSN1994I DSN1COMP COMPLETED SUCCESSFULLY, 38 PAGES PROCESSED

Chapter 43. DSN1COMP  959
Chapter 44. DSN1COPY

You can use the DSN1COPY stand-alone utility to copy Db2 VSAM data sets.

With the DSN1COPY stand-alone utility, you can copy:
- Db2 VSAM data sets to sequential data sets
- DSN1COPY sequential data sets to Db2 VSAM data sets
- Db2 image copy data sets to Db2 VSAM data sets
- Db2 VSAM data sets to other Db2 VSAM data sets
- DSN1COPY sequential data sets to other sequential data sets

A Db2 VSAM data set is one of the following items:
- A single piece of a nonpartitioned table space or index
- A single partition of a partitioned table space or index
- A FlashCopy image copy data set

The input must be a single z/OS sequential or VSAM data set. Concatenation of input data sets is not supported.

You can also use DSN1COPY to perform the following actions:
- Print hexadecimal dumps of Db2 data sets and databases
- Check the validity of data or index pages (including dictionary pages for compressed data)
- Translate database object identifiers (OBIDs) to enable moving data sets between different systems
- Reset to 0 the log RBA that is recorded in each index page or data page.

You cannot run DSN1COPY on concurrent copies.

DSN1COPY can operate on both base and clone objects.

In most cases, specify a VSAM cluster rather than a specific data component. The only exception is when you use DSN1COPY for a specific data component. (For example, specify the data component when printing or copying that specific component and not the entire cluster.) In the following data set name format, the bold "C" indicates a cluster. A "D" in that place of the data set name indicates a data component.

```
catname.DSNDBC.dbname.psname.y001.z001
```

You can use the DSN1COPY utility on LOB table spaces by specifying the LOB keyword and omitting the SEGMENT and INLCOPY keywords.

Db2-managed data sets can be moved from hard disk drives to solid state drives by using DSN1COPY.

Output

The first time that the CHECK INDEX, CHECK DATA, COPY, REBUILD INDEX, REORG TABLESPACE, RUNSTATS, or UNLOAD utility physically opens a data set after DSN1COPY populates the data set, Db2 checks for any data and catalog inconsistencies. Those checks are the same ones that REPAIR CATALOG performs.
The first time that an SQL data manipulation statement or query physically opens a data set after DSN1COPY populates the data set, Db2 checks for any data and catalog inconsistencies for the following items:

- DBID, PSID, and OBID
- Attributes that are defined by SEGSIZE and BUFFERPOOL (page size) options
- Table space type
- Table schema
  
  Db2 checks this item only if the table space contains a system page for the table.

Db2 reports any inconsistencies with a -904 SQL code, and you cannot access the data.

Db2 does not check these items for LOB or XML table spaces or for index spaces. Db2 also does not validate record row format or RBA format.

Db2 does not check for data and catalog inconsistencies during the following situations:

- Db2 is restarting.
- The header page is not formatted yet.
- The REPAIR utility is operating on the header page. (The REPAIR utility closes the page set when it is finished. Therefore, validation can be done the next time that the data set is physically opened.)
- The LOGAPPLY phase of the RECOVER utility is running.
- The LOAD utility is running.

By not checking for inconsistencies during these situations, Db2 limits any performance impact.

You can correct some of the reported inconsistencies by using the REPAIR utility with the CATALOG option. The REPAIR CATALOG output indicates which inconsistencies it can fix.

Related information:

- Table space types and characteristics in Db2 for z/OS (Introduction to Db2 for z/OS)
- “Syntax and options of the REPAIR control statement” on page 670

Environment

Run DSN1COPY as a z/OS job when the Db2 subsystem is either active or not active.

If you run DSN1COPY when Db2 is active, use the following procedure:

1. Start the source table space in read-only mode by issuing the -START DATABASE command for the table space.
2. Run the QUIESCE utility with the WRITE (YES) option on the source table space to externalize all data pages and index pages.
3. Run DSN1COPY.
4. Start the source table space in read/write mode by issuing the -START DATABASE command.
Authorization required

DSN1COPY does not require authorization. However, if any of the data sets is protected by RACF, the authorization ID of the job must have RACF authority.

Restrictions

DSN1COPY does not alter data set structure. For example, DSN1COPY does not convert a partitioned or segmented table space to a simple table space if you copy the contents of a partitioned or segmented table space to a simple table space. The output data set is a page-for-page copy of the input data set. If the intended use of DSN1COPY is to move or restore data, ensure that definitions for the source and target table spaces, tables, and indexes are identical. Otherwise, unpredictable results can occur.

DSN1COPY cannot copy Db2 recovery log data sets. The format of a Db2 log page is different from the format of a table or index page. If you try to use DSN1COPY to recover log data sets, DSN1COPY abnormally terminates.

All the target data sets must exist. You can use Access Method Services to define them.

For a compressed table space, DSN1COPY does not reset the dictionary version for the following items:

- An inline image copy
- An incremental image copy that was created with the SYSTEMPAGES=YES COPY utility option

To reset the dictionary version for an inline image copy, use the inline image copy as input to DSN1COPY with a VSAM intermediate data set as output. This intermediate data set can then be used as input to DSN1COPY RESET to copy the intermediate data set to the real target data set.

DSN1COPY issues an error and terminates in the following situations:

- DSN1COPY can verify that the LOB option is specified, but the data set is not a LOB data set.
- The LOB option is omitted for a data set that is a LOB data set.

To avoid problems, always specify the LOB option if the input data set SYSUT1 is a LOB table space, and make sure that the LOB option is not specified for non LOB table spaces.

DSN1COPY cannot copy a source object of 4 GB or greater in size when it is full unless the target object is EA-enabled. For example, the source is full when it is not the last piece of a multi-piece non-partitioned object with a DSSIZE of 4 GB or greater. To avoid VSAM errors and limit each piece to 2 GB so that the target object has more pieces than the original source:

- Define the target data set as EA-enabled and use DSN1COPY to copy the data, one piece at a time, from the source that is not EA-enabled to the target.
- If it is not possible to define the target data set as EA-enabled:
  1. Take a full image copy of the entire source object by running the COPY utility and specifying DSNUM ALL.
  2. Create the target object by specifying DSSIZE 2GB for table spaces and PIECESIZE 2GB for indexes. See "Copying tables from one subsystem to another" on page 981.
3. Define the partition number data sets (2 GB each) with the IDCAMS command. Define enough pieces to hold the entire source.

4. Run the DSN1COPY utility with the image copy as the source (SYSUT1), the target object as SYSUT2, and specify DSSIZE 2G.

DSN1COPY cannot be used to restore data to a point in time before materialization of pending definition changes.

For partition-by-growth table spaces, DSN1COPY can be used only if the number of active partitions of the source and the target table space are the same. If the number of partitions in the source table space and the target table space are not the same, take one of the following actions:

- If the number of partitions in the source table space is greater than the number of partitions in the target table space, issue SQL statement ALTER TABLE with the ADD PARTITION option on the table in the target table space to make the number of partitions the same in the source and target table spaces. Then use DSN1COPY to copy the contents of the source table space to the target table space.

- Use the UNLOAD utility to unload the data from the source table space, and use the LOAD utility to reload the data into the target table space.

**Syntax and options of the DSN1COPY control statement**

The DSN1COPY utility control statement, with its multiple options, defines the function that the utility job performs.
**Option descriptions**

To run DSN1COPY with invocation parameters, specify one or more of the following parameters on the EXEC statement. If you specify more than one parameter, separate each parameter by a comma. You can specify parameters in any order.

Default settings for DSN1COPY options are taken from the input data set header page. This default processing is recommended when running DSN1COPY because incorrect parameter settings can result in unpredictable results.

When non-default user values are specified, DSN1COPY compares the input data set header page settings against user-specified values whenever possible. If a mismatch is detected, message DSN1930I is issued. The processing is performed with the user-specified values.

**CHECK**

Checks each page from the SYSUT1 data set for validity. The validity checking operates on one page at a time and does not include any cross-page checking. If an error is found, a message is issued describing the type of error, and a dump of the page is sent to the SYSPRINT data set. If an unexpected page number is encountered, validity checking continues to the end and a report will be printed of all unexpected page numbers. If you do not receive any messages, no errors were found. If more than one error exists in a given page,
the check identifies only the first of the errors. However, the entire page is
dumped. DSN1COPY does not check system pages for validity.

An index with BUSINESS_TIME period columns appended to the key for
BUSINESS TIME WITHOUT OVERLAPS bypasses checking for orderly keys.

**32K**

Specifies that the SYSUT1 data set has a 32-KB page size. If you specify this
option and the SYSUT1 data set does not have a 32-KB page size, DSN1COPY
might produce unpredictable results that might be undetected until later.

**PAGESIZE**

Specifies the page size of the input data set that is defined by SYSUT1.
Available page size values are 4K, 8K, 16K, or 32K. If you specify an incorrect
page size, DSN1COPY might produce unpredictable results.

If you do not specify the page size, DSN1COPY tries to determine the page
size from the input data set if the first page of the input data set is a header
page. Db2 issues an error message if DSN1COPY cannot determine the input
page size. This might happen if the header page is not in the input data set, or
if the page size field in the header page contains an invalid page size.

**FULLCOPY**

Specifies that a Db2 full image copy (not a DFSMSdss concurrent copy) of your
data is to be used as input. If this data is partitioned, specify NUMPARTS to
identify the total number of partitions. If you specify FULLCOPY without
NUMPARTS, DSN1COPY determines the NUMPARTS value from the header
page if possible; otherwise, DSN1COPY assumes that your input file is not
partitioned.

Specify FULLCOPY when using a full image copy as input. Omitting the
parameter can cause error messages or unpredictable results.

Do not specify FULLCOPY if you are using a FlashCopy image copy data set
as input.

The FULLCOPY parameter requires SYSUT2 (output data set) to be either a
Db2 VSAM data set or a DUMMY data set.

**INCRCOPY**

Specifies that an incremental image copy of the data is to be used as input.
DSN1COPY with the INCRCOPY parameter updates existing data sets; do not
redefine the existing data sets. INCRCOPY requires that the output data set
(SYSUT2) be a Db2 VSAM data set.

Before you apply an incremental image copy to your data set, you must first
apply a full image copy to the data set by using the FULLCOPY parameter.
Make sure that you apply the full image copy in a separate execution step
because you receive an error message if you specify both the FULLCOPY and
the INCRCOPY parameters in the same step. Then, apply each incremental
image copy in a separate step, starting with the oldest incremental image copy.

Specifying neither FULLCOPY nor INCRCOPY implies that the input is not an
image copy data set. Therefore, only a single output data set is used.

**SEGMENT**

Specifies that you want to use a segmented table space as input to DSN1COPY.
Pages with all zeros in the table space are copied, but no error messages are
issued. You cannot specify FULLCOPY or INCRCOPY if you specify
SEGMENT.
If you are using DSN1COPY with the OBIDXLAT to copy a Db2 data set to another Db2 data set, the source and target table spaces must have the same SEGSIZE attribute.

You cannot specify the SEGMENT option with the LOB parameter.

**INLCOPY**

Specifies that the input data is an inline copy data set. The INLCOPY parameter requires SYSUT2 (output data set) to be either a VSAM data set or a DUMMY data set.

You cannot specify the INLCOPY option with the LOB parameter.

**DSSIZE(integer G)**

Specifies the data set size, in gigabytes, for the input data set. If you omit DSSIZE, Db2 obtains the data set size from the data set header page.

If you specify DSSIZE, integer must match the DSSIZE value that was specified when the table space was defined.

**LARGE**

Specifies that the input data set is a table space that was defined with the LARGE option, or an index on such a table space. If you specify the LARGE keyword, Db2 assumes that the data set has a 4-GB boundary. The recommended method of specifying a table space that was defined with the LARGE option is DSSIZE(4G). If you omit the LARGE or DSSIZE(4G) option when it is needed, or if you specify LARGE for a table space that was not defined with the LARGE option, the results from DSN1COPY are unpredictable.

If you specify LARGE, you cannot specify LOB or DSSIZE.

**LOB**

Specifies that SYSUT1 data set is a LOB table space. Empty pages in the table space are copied, but no error messages are issued. You cannot specify the SEGMENT and INLCOPY options with the LOB parameter.

DSN1COPY attempts to determine if the input data set is a LOB data set. If it can be clearly verified that the LOB option is specified, but the data set is not a LOB data set, or that the LOB option is omitted for a data set that is a LOB data set, DSN1COPY issues an error message and terminates. Otherwise, if the LOB option isn't specified or omitted correctly the results of DSN1COPY are unpredictable.

If you specify LOB, you cannot specify LARGE.

**NUMPARTS(integer)**

Specifies the number of partitions that are associated with the input data set. Valid specifications range 1 - 4096. If you omit NUMPARTS or specify it as 0, DSN1COPY assumes that your input file is not partitioned. If you specify a number greater than 64, DSN1COPY assumes that the data set is for a partitioned table space that was defined with the LARGE option, even if the LARGE keyword is not specified.

DSN1COPY cannot always validate the NUMPARTS parameter. If you specify it incorrectly, DSN1COPY might produce unpredictable results.

DSN1COPY terminates and issues message DSN1946I when it encounters an image copy that contains multiple partitions; a compression report is issued for the first partition.
This parameter is not used if the target table space is a universal table space. DSSIZE is used instead.

This parameter is deprecated.

PRINT(hexadecimal-constant,hexadecimal-constant)
Causes the SYSUT1 data set to be printed in hexadecimal format on the SYSPRINT data set. You can specify the PRINT parameter with or without the page range specifications (hexadecimal-constant,hexadecimal-constant). If you do not specify a range, all pages of the SYSUT1 are printed. If you want to limit the range of pages that are printed, indicate the beginning and ending page. If you want to print a single page, supply only that page number. In either case, your range specifications must be from one to eight hexadecimal characters in length.

The following example shows how you code the PRINT parameter if you want to begin printing at page X'2F0' and stop at page X'35C':
PRINT(2F0,35C)

Because the CHECK and RESET options and the copy function run independently of the PRINT range, these options apply to the entire input file, regardless of whether a range of pages is being printed.

You can indicate the format of the row data in the PRINT output by specifying EBCDIC, ASCII, or UNICODE.

EBCDIC
Indicates that the row data in the PRINT output is to be displayed in EBCDIC. The default value is EBCDIC if the first page of the input data set is not a header page.

If the first page is a header page, DSN1COPY uses the format information in the header page as the default format. However, if you specify EBCDIC, ASCII, or UNICODE, that format overrides the format information in the header page. The unformatted header page dump is always displayed in EBCDIC, because most of the fields are in EBCDIC.

ASCII
Indicates that the row data in the PRINT output is to be displayed in ASCII. Specify ASCII when printing table spaces that contain ASCII data.

UNICODE
Indicates that the row data in the PRINT output is to be displayed in Unicode. Specify UNICODE when printing table spaces that contain Unicode data.

PIECESIZ(integer)
Specifies the maximum piece size (data set size) for nonpartitioned indexes. The value that you specify must match the value that was specified when the nonpartitioning index was created or altered. The defaults for PIECESIZ are 2G (2 GB) for indexes that are backed by non-large table spaces and 4G (4 GB) for indexes that are backed by table spaces that were defined with the LARGE option. This option is required if the piece size is not one of the default values. If PIECESIZ is omitted and the index is backed by a table space that was defined with the LARGE option, the LARGE option is required for DSN1COPY.

The subsequent keyword K, M, or G indicates the unit of the value that is specified in integer.
**K** Indicates that the *integer* value is to be multiplied by 1 KB to specify the maximum piece size in bytes. *integer* must be either 256 or 512.

**M** Indicates that the *integer* value is to be multiplied by 1 MB to specify the maximum piece size in bytes. *integer* must be a power of two, between 1 and 512.

**G** Indicates that the *integer* value is to be multiplied by 1 GB to specify the maximum piece size in bytes. *integer* must be a power of two, between 1 and 256.

Valid values for piece size are:
- 1 MB or 1 GB
- 2 MB or 2 GB
- 4 MB or 4 GB
- 8 MB or 8 GB
- 16 MB or 16 GB
- 32 MB or 32 GB
- 64 MB or 64 GB
- 128 MB or 128 GB
- 256 KB, 256 MB, or 256 GB
- 512 KB or 512 MB

**VALUE**
Causes each page of the SYSUT1 input data set to be scanned for the character string that you specify in parentheses following the VALUE parameter. Each page that contains that character string is printed in the SYSPRINT data set. You can specify the VALUE parameter in conjunction with any of the other DSN1COPY parameters.

*string* can consist of 1 to 20 alphanumeric characters.

*hexadecimal-constant* can consist of 2 to 40 hexadecimal characters. Specify two apostrophe characters before and after the hexadecimal character string.

If you want to search your input file for the string '12345', your JCL should look similar to the following JCL:
```
//STEP1 EXEC PGM=DSN1COPY,PARM='VALUE(12345)'
```

Alternatively, you might want to search for the equivalent hexadecimal character string. If you are processing Unicode or ASCII input files, you must specify the string in hexadecimal. Your JCL should look similar to the following JCL:
```
//STEP1 EXEC PGM=DSN1COPY,PARM='VALUE(''3132333435'')'
```

**OBDXLAT**
Specifies that OBID translation must be done before the Db2 data set is copied. OBID translation is needed when the source and target OBIDs do not match. This parameter requires additional input from the SYSXLAT file by using the DD statements. DSN1COPY can translate only up to 10000 record OBIDs.

If you specify OBDXLAT, CHECK processing is performed, regardless of whether you specify the CHECK option.

**Related information:**
- “The effects of not specifying the OBDXLAT option” on page 977

**RESET**
Causes the log RBAs in each index page or data page and the high-formatted
page number in the header page to be reset to 0. If you specify this option, CHECK processing is performed, regardless of whether you specify the CHECK option.

Use RESET when the output file is used to build a Db2 table space that is to be processed on a Db2 subsystem with a different recovery log than the source subsystem. Failure to specify RESET in such a case can result in an abend during subsequent update activity. The abend reason code of 00C200C1 indicates that the specified RBA value is outside the valid range of the recovery log. A condition code of 0 indicates successful completion.

Do not specify the RESET parameter for page sets that are in group buffer pool RECOVER-pending (GRECP) status.

For a compressed table space, DSN1COPY does not reset the dictionary version for an inline image copy, or for an incremental image copy that was created with the SYSTMPAGES=YES COPY utility option.

If you do not specify RESET when copying a table space from one Db2 system to another, a down-level ID check might result in abend reason code 00C2010D when the table space is accessed.

Related information:

Recovering from a down-level page set problem (Db2 Administration Guide)

Before running DSN1COPY

Certain activities might be required before you run the DSN1COPY utility, depending on your situation.

Attention: Do not use DSN1COPY in place of COPY for both backup and recovery. Improper use of DSN1COPY can result in unrecoverable damage and loss of data.

Recommendations

Printing with DSN1PRNT instead of DSN1COPY

If you require only a printed hexadecimal dump of a data set, use DSN1PRNT rather than DSN1COPY.

Copying a table space with DSN1COPY with row formats

When you use a DSN1COPY of a table space to populate another table space, the row formats of the two table spaces must match. If the row formats do not match, the results are unpredictable and could cause integrity problems.

To determine the source table space and target table space row format, run the following query against your Db2 catalog:

```sql
SELECT DBNAME, TSNAMES, PARTITION, FORMAT
FROM SYSIBM.SYSTABLEPART
WHERE (DBNAME = 'source-database-name'
AND TSNAMES='source-table-space-name')
OR (DBNAME = 'target-database-name'
AND TSNAMES='target-table-space-name')
```

If the FORMAT column has a value of 'R', then the table space or partition is in RRF (reordered row format). If the FORMAT column has a blank value, then the table space or partition is in BRF (basic row format).

Determining page size and DSSIZE
Before using DSN1COPY, ensure that you know the page size and data set size (DSSIZE) for the page set. Use the following query on the Db2 catalog to get the information you need in this example for table 'DEPT':

```sql
SELECT T.CREATOR, T.NAME, S.NAME AS TABLESPACE, S.PARTITIONS, S.PGSIZE,
CASE S.DSSIZE
  WHEN 0 THEN
    CASE WHEN S.TYPE = 'G' THEN 4194304
         WHEN S.TYPE = 'O' THEN 4194304
         WHEN S.TYPE = 'P' THEN 4194304
         WHEN S.TYPE = 'R' THEN 4194304
    ELSE
      CASE WHEN S.PARTITIONS > 254 THEN
        CASE WHEN S.PGSIZE = 4 THEN 4194304
             WHEN S.PGSIZE = 8 THEN 8388608
             WHEN S.PGSIZE = 16 THEN 16777216
             WHEN S.PGSIZE = 32 THEN 33554432
       ELSE NULL
       END
       WHEN S.PARTITIONS > 64 THEN 4194304
       WHEN S.PARTITIONS > 32 THEN 1048576
       WHEN S.PARTITIONS > 16 THEN 2097152
       WHEN S.PARTITIONS = 0 THEN 4194304
       ELSE 2097152
     END
   END
 ELSE S.DSSIZE
END AS DSSIZE
FROM SYSIBM.SYSTABLES T,
     SYSIBM.SYSTABLESPACE S
WHERE T.NAME = 'DEPT' AND
   T.TSNAME = S.NAME;
```

*Figure 106. Example catalog query that returns the page set size and data set size for the page set.*

**Using the OBIDXLAT option with DSN1COPY**

When you use DSN1COPY with the OBIDXLAT option to move objects from one system to another system, ensure that the version information on the target system matches the version information on the source version.

**Copying a partition-by-range or partition-by-growth table space**

When you use DSN1COPY on a partition-by-range or partition-by-growth space, use the SEGMENT and NUMPARTS options to process the table space. For partition-by-growth table spaces, the NUMPART value specified should be the MAXPARTITIONS value that the table space was created with.

**Copying when pending alterations exist**

Before you use DSN1COPY, ensure that the schema of the source and target objects match.

You might also need to run the REORG TABLESPACE utility to materialize pending alterations depending on the following conditions:

- If the pending alterations are for an added or dropped column, run
  ```sql
  REORG TABLESPACE.
  ```
- If the pending alteration are for a changed data type, you need to either insert or update at least one row or run REORG TABLESPACE.

After you run DSN1COPY, run REPAIR CATALOG.
Related information:

“Syntax and options of the REPAIR control statement” on page 670

Copying a versioned XML table space

Before using DSN1COPY to copy a versioned XML table space, ensure that the definitions of the XML columns START_TS and END_TS match.

Altering a table

When you use ALTER TABLE ADD COLUMN, the table does not change; only the description of the table changes. Before you run DSN1COPY on the table space, run REORG on the table space (so that the data matches its description).

Related concepts:

Table space versions (Db2 Administration Guide)

Related tasks:

“Copying tables from one subsystem to another” on page 981

Data sets that DSN1COPY uses

The DSN1COPY utility uses a number of data sets during its operation.

Required data sets

DSN1COPY uses the following data sets:

Input data set

Input to DSN1COPY. The DD name is SYSUT1.

Output data set

Output from DSN1COPY. The DD name is SYSUT2. Optional.

Message data set

Data set for output messages. The DD name is SYSPRINT.

OBIDXLAT data set

Data set that defines the OBID translation values. The DD name is SYSXLAT.

DSN1COPY uses the following DD statements:

SYSPRINT

Defines the data set that contains output messages from the DSN1COPY program and all hexadecimal dump output.

SYSUT1

Defines the input data set. This data set can be a sequential data set that is created by the DSN1COPY or COPY utilities, or a VSAM data set, including a FlashCopy image copy data set.

Specify the data set’s disposition as DISP=OLD to ensure that it is not in use by Db2. Specify the data set’s disposition as DISP=SHR only when the Db2 STOP DATABASE command does not work.

The requested operation takes place only for the specified data set. If the input data set is a partitioned table space or index, ensure that you specify the NUMPARTS parameter and the correct data set. For example, to print a page range in the second partition of a four-partition table space, specify NUMPARTS(4) and the data set name of the second data set. This second
data set is in the group of VSAM data sets, and the VSAM data set name is DSNCAT.DSNDBD.TESTDB.TS01.I0001.A002. The last qualifier (A002) represents the partition number 2.

To copy the data sets for a segmented table space that consists of multiple data sets, you need to run multiple DSN1COPY jobs. Run one job for each data set in the table space.

If running the online REORG utility with the FASTSWITCH option, verify the data set name before running the DSN1COPY utility. The fifth-level qualifier in the data set name alternates between I0001 and J0001 when using FASTSWITCH. If the table space has cloning or ever had cloning, the fifth-level qualifier can be I0002 or J0002. You cannot specify FASTSWITCH YES if the table space has cloning; however, a FASTSWITCH YES REORG might have been done before the clone was created so you might still have a mixture of 'I' and 'J' data sets. Specify the correct fifth-level qualifier in the data set name to successfully execute the DSN1COPY utility. To determine the correct fifth-level qualifier, query the IPREFIX column of SYSIBM.SYSTABLEPART for each data partition or the IPREFIX column of SYSIBM.SYSINDEXPART for each index partition. If the object is not partitioned, use zero as the value for the PARTITION column in your query.

To determine the instance number to use for a fifth-level qualifier, query the INSTANCE column of SYSIBM.SYSTABLESPACE. The returned value is the instance number that represents the current base objects. The clone objects would be represented by the other instance number. If a query of SYSTABLESPACE.INSTANCE returns a value of 2, then the base objects are represented by instance number 2 data sets and the clone objects by instance number 1 data sets. This process can be used to determine the instance number even if there is no active cloning.

SYSUT2

Defines the output data set. This data set can be a sequential data set, a VSAM data set, or a DUMMY data set.

What you specify for SYSUT2 is restricted if both of the following conditions are true:

• SYSUT1 is an image copy of an entire partitioned table space or an image copy of all data sets of a multi-piece object
• The data is to be copied to a Db2 table space or Db2 index space

In this case, SYSUT2 must be the name of first data set (of the first partition or of the first piece) for the table space or index space. For example, the last part of the name must identify the data set number A001.

DSN1COPY identifies the appropriate output data set by the page number and allocates other data sets for additional partitions. The names of these data sets also follow the Db2 data set naming conventions. For example, these data set names end with A002, A003, and so on.

All target data sets must be defined. To define the data sets for a multi-piece object, create first the table space or index with DEFINE YES and specify appropriate primary and secondary quantities. If -1 is specified for primary and/or secondary quantity DSN1COPY may run out of extents because Db2 defines the first data set with small primary and/or secondary extents in this case. Db2 then defines the first data set. The
subsequent data sets can be defined by using Access Method Services. See “Copying tables from one subsystem to another” on page 981 for more information.

DSN1COPY assumes that the output data sets are empty (that is, the program adds the blocks) except when you specify INCRCOPY. Before you run DSN1COPY, define your VSAM output data sets as REUSE. If you have not defined the data sets, you must redefine all VSAM output data sets you are restoring by using Access Method Services. Ensure that these data sets are empty before you run DSN1COPY.

You might want to specify a DUMMY SYSUT2 DD statement if you are dumping or checking pages.

To enable Db2 to obtain necessary information from the integrated catalog facility catalog when using VSAM data sets, do not code the unit-serial parameter and volume-serial parameter.

If running the online REORG utility with the FASTSWITCH option, verify the data set name before running the DSN1COPY utility. The fifth-level qualifier in the data set name alternates between I0001 and J0001 when using FASTSWITCH. Specify the correct fifth-level qualifier in the data set name to successfully execute the DSN1COPY utility.

SYSLXTA

Defines for translation the DBIDs, OBIDs, data page set identifiers (PSIDs), or index page set identifiers (ISOBIDs).

If you have dropped a table without a subsequent REORG of the table space, you must reorganize the source table space before running DSN1COPY with the OBIDXLAT option. This action removes any previously dropped records from the table space.

A non-numeric character must separate each record in the SYSLXTA file, and each record must contain a pair of decimal integers. The first integer of each record pertains to the source, and the second integer pertains to the target. The first record in the SYSLXTA file contains the source DBIDs and the target DBIDs; the values can range from -32767 to 65535. The second record contains the source and target PSIDs or ISOBIDs; the values can range from 0 to 32767. All subsequent records in the SYSLXTA data set are for table OBIDs. For an index, the SYSLXTA data set must contain the index fan set OBID, in addition to the DBID and ISOBID. Sample data in a SYSLXTA file follows (with an indication of how each record translates shown in parentheses):

<table>
<thead>
<tr>
<th>Source OBID</th>
<th>Target OBID</th>
</tr>
</thead>
<tbody>
<tr>
<td>260</td>
<td>280</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>55</td>
</tr>
<tr>
<td>6</td>
<td>56</td>
</tr>
<tr>
<td>7</td>
<td>57</td>
</tr>
</tbody>
</table>

To obtain the names, DBIDs, PSIDs, ISOBIDs, and OBIDs, run the DSNTEP2 sample application on both the source and target systems. The following SQL statements yield the preceding information.

The example for indexes yields output that is similar to the preceding example, but with an additional column of data.

For table spaces use the following statements:
SELECT DBID, PSID FROM SYSIBM.SYSTABLESPACE
   WHERE NAME='tablespace_name'
   AND DBNAME='database_name';
SELECT NAME, OBID FROM SYSIBM.SYSTABLES
   WHERE TSNAME='tablespace_name'
   AND CREATOR='creator_name';

For index spaces use the following statement:
SELECT DBID, ISOBID, OBID FROM SYSIBM.SYSINDEXES
   WHERE NAME='index_name'
   AND CREATOR='creator_name';

Several examples of using DSN1COPY follow:
• Create a backup copy of a Db2 data set:
  – SYSUT1: DB2-VSAM
  – SYSUT2: Sequential data set
• Restore a backup copy of a Db2 data set:
  – SYSUT1: DSN1COPY sequential data set
  – SYSUT2: DB2-VSAM
• Move a Db2 data set to another Db2 data set:
  – SYSUT1: DB2-VSAM
  – SYSUT2: DB2-VSAM
  – Parameters: OBIDXLAT, RESET
• Perform validity checking on a Db2 data set:
  – SYSUT1: DB2-VSAM
  – SYSUT2: DUMMY
  – Parameter: CHECK
• Perform validity checking on and print a Db2 data set:
  – SYSUT1: DB2-VSAM
  – SYSUT2: DUMMY
  – Parameters: CHECK, PRINT
• Restore a table space from a nonpartitioned image copy data set or page set:
  – SYSUT1: Db2 full image copy
  – SYSUT2: DB2-VSAM
  – Parameter: FULLCOPY
• Restore a table space from a partitioned image copy data set or page set:
  – SYSUT1: Db2 full image copy
  – SYSUT2: DB2-VSAM
  – Parameters: FULLCOPY, NUMPARTS(nn)
• Perform RBA RESET on a Db2 data set:
  – SYSUT1: DB2-VSAM or DSN1COPY sequential data set
  – SYSUT2: DB2-VSAM
  – Parameter: RESET

**Defining the input data set**

The SYSUT1 data set can be any of the following types:
• A Db2 table space data set
• A Db2 index space data set
• A sequential full image copy
• An incremental image copy
• An inline image copy
A sequential data set that was previously created by DSN1COPY
A FlashCopy image copy data set

Define SYSUT1 with DISP=OLD to ensure that DSN1COPY uses it exclusively. If SYSUT1 is a table space or index space, use the following procedure before using DSN1COPY:

1. Issue the following command to determine if the object is stopped:
   -DISP=DATABASE (database_name)  SPACENAM(space_name)  RESTRICT
2. If Db2 has not stopped the object, issue the following command to stop the object:
   -STOP DATABASE (database_name)  SPACENAME(space_name)

Db2 allows input of only one DSN1COPY data set. Db2 does not permit the input of concatenated data sets. For a table space that consists of multiple data sets, ensure that you specify the correct data set. For example, if you specify the CHECK option to validate pages of a partitioned table space's second partition, code the second data set of the table space for SYSUT1.

**Defining the output data set**

The SYSUT2 data set can be any of the following types:
- A sequential data set
- A Db2 table space data set
- A Db2 index space data set
- A DUMMY data set

Specify a DUMMY SYSUT2 DD statement if you are using DSN1COPY to check or dump a page. The table spaces and index spaces must either be empty or defined with VSAM REUSE. STOGROUP-defined table spaces and index spaces have the REUSE attribute, except when you are applying the INCRCOPY option.

When you use the RESET option to reset page log RBAs, you need to ensure that the output data set for the RESET operation has the same name as the original Db2 data set. Use either of the following techniques to do that:

- Method 1:
  1. Make a backup copy of your original Db2 data set by using DSN1COPY to copy the original data set to a sequential data set.
  2. If you defined your original Db2 data set without the REUSE parameter, delete and redefine the original data set.
  3. Run DSN1COPY with the RESET option. Specify the output data set from step 1 as the input data set for the RESET operation.

- Method 2:
1. Run DSN1COPY with the RESET option. Use your original Db2 data set as the input data set. Define a new VSAM data set as the output data set. The output data set must have the same data set characteristics as the input data set.

2. Delete the input data set from step 1.

3. Rename the output data set from step 1 to the same name as the input data set.

**Adding additional volumes for SYSUT2**

When you create a table space or index space by using STOGROUP, the ICF catalog entry has only one volume in the volume list. If the SYSUT2 data set that DSN1COPY restores requires more than one volume, use the IDCAMS command, ALTER ADDVOLUMES, to add additional volume IDs to the integrated catalog entry. The extension to new volumes uses the primary size on each new volume. This is the normal VSAM extension process. If you want the data set to use the secondary size on the candidate volumes, follow these steps:

1. Run DSN1COPY.
2. Run REORG, or make a full image copy and recover the table space.

Performing these steps resets the data set and causes normal extensions through Db2.

**Related reference:**


**Inconsistent data checks**

When critical data is involved, use the CHECK option of DSN1COPY to prevent the undetected copying of inconsistent data to the output data set. The CHECK option performs validity checking on one page at a time.

You must run a CHECK utility job on the table space that is involved to ensure that no inconsistencies exist between data and indexes on that data:

- Before using DSN1COPY to save critical data that is indexed
- After using DSN1COPY to restore critical data that is indexed

The CHECK utility performs validity checking between pages.

**Tip:** For best results, check the consistency of the Db2 catalog and directory regularly, even outside of the migration process. For detailed instructions, see Verify the integrity of Db2 table spaces and Check for consistency between catalog tables.

**The effects of not specifying the OBIDXLAT option**

If you use DSN1COPY to load data into a table space or index without specifying the OBIDXLAT option, be careful not to invalidate embedded Db2 internal identifiers.

Those OBIDs can become invalid in the following circumstances:

- When you drop and re-create tables after the input data set to DSN1COPY was created.
- When a difference exists among the following attributes between the target subsystem and the source subsystem:
- Table space attributes of BUFFERPOOL or NUMPARTS
- Table attributes other than table name, table space name, and database name
- The order of the table spaces, indexes, and tables that the user defined or dropped in the source and target databases

To protect against invalidating the OBIDs, specify the OBIDXLAT parameter for DSN1COPY. The OBIDXLAT parameter translates OBID, DBID, PSID, or ISOBID before DSN1COPY copies the data.

### Requirements for using an image copy as input to DSN1COPY

To use image copies (full sequential or incremental) as input to DSN1COPY, you must use the COPY utility with SHRLEVEL REFERENCE to produce those image copies.

Using the FULLCOPY parameter ensures that the data that is contained in your image copies is consistent. DSN1COPY accepts an index image copy as input when you specify the FULLCOPY option. If you want to use inline image copies as input to DSN1COPY, you must produce those image copies by using the REORG utility or LOAD utility.

If you want to use a FlashCopy image copy data set as input, do not specify the FULLCOPY option.

### Copying from an image copy

You can use DSN1COPY to copy data from an image copy of the data sets of a table space to the data sets of a table space on the same subsystem or another subsystem.

#### Procedure

To copy from a specific image copy data set, specify the following SYSUT2 data sets:

- **If SYSUT1 is an image copy of a single partition**, ensure that the SYSUT2 DD statement refers to the first data set of the table space. DSN1COPY determines the correct target data set. Code the NUMPARTS(nn) parameter, where nn is the number of partitions in the entire table space. However, if the partitioned table space is defined with more than one VCAT name (for example, a unique VCAT for different partitions), use SYSUT2 as the name of the data set for that partition.

- **If SYSUT1 is an image copy of an entire partitioned table space**, ensure that the SYSUT2 DD statement refers to the first data set of the table space. In this case, DSN1COPY allocates all of the target data sets. However, you must have previously defined the target data sets either by creating the partitioned table space with DEFINE YES or by using Access Method Services. Code the NUMPARTS parameter as described in the first bullet when the table space is partitioned. When multiple VCAT names are used for different partitions of a partitioned table space, DSN1COPY cannot restore the entire table space by using as input a single full image copy of the table space. In this case, when you use DSN1COPY, you must restore individual copies of each partition by using the name of the data sets for that partition. Code the NUMPARTS(nn) parameter, where nn is the number of partitions in the entire table space.

- **If SYSUT1 is an image copy of a single data set of a multiple data set nonpartitioned table space**, ensure that the SYSUT2 DD statement refers to the
actual (not the first) output data set of the table space. Do not specify NUMPARTS because this parameter is only for partitioned table spaces.

- **If SYSUT1 is an image copy of an entire multiple data set nonpartitioned table space**, ensure that the SYSUT2 DD statement refers to the first data set of the table space. DSN1COPY allocates all target data sets. However, you must have previously defined the target data sets by using Access Method Services. If the source data sets are less than the target data sets it is recommended to either delete all rows from the target table space or to do a LOAD REPLACE with DD DUMMY on the target table space first, before running DSN1COPY.

- **If SYSUT1 is an image copy of a single data set of a multiple data set LOB table space**, ensure that the SYSUT2 DD statement refers to the actual (not the first) data set of the table space. Do not specify NUMPARTS because this parameter is only for partitioned table spaces.

- **If SYSUT1 is an image copy of an entire multiple data set LOB table space**, ensure that the SYSUT2 DD statement refers to the first data set of the table space. DSN1COPY allocates all target data sets. However, you must have previously defined the target data sets by using Access Method Services.

**What to do next**

**Important**: After you use DSN1COPY to copy data from an image copy, you need to ensure that the version information in the source and target table spaces matches. To do that, run REPAIR CATALOG on the target table space immediately after you run DSN1COPY.

**Related tasks**:

- "Copying tables from one subsystem to another" on page 981

---

**Restoring indexes with DSN1COPY**

When a table space is restored using either the TOCOPY option of RECOVER or the DSN1COPY utility, restore the indexes.

**Procedure**

To restore indexes with DSN1COPY, use one of the following methods:

- Use the RECOVER utility, if you have a full image copy available, and the index was defined with the COPY YES option.

- Use DSN1COPY on the indexes, if a copy is available. If you specified the OBIDXLAT option for the data, you must also specify the OBIDXLAT option for the indexes. Also, the indexes must all have been copied at the same time as the data; otherwise, inconsistencies might exist.

- If you do not have an image copy of the index, use the REBUILD INDEX utility, which reconstructs the indexes from the data.

**Related concepts**:

- “The effects of not specifying the OBIDXLAT option” on page 977

**Related reference**:

- Chapter 25, “REBUILD INDEX,” on page 427
- Chapter 26, “RECOVER,” on page 459
Restoring table spaces with DSN1COPY

In certain cases, you cannot use the RECOVER utility for an image copy data set. In these cases, consider using the DSN1COPY utility to restore the table space or data set instead.

**About this task**

You cannot use RECOVER TOCOPY for an image copy data set that is not referenced by SYSIBM.SYSCOPY for that table space or data set. An attempt to do so results in message DSNU519I "TOCOPY DATASET NOT FOUND". The MODIFY utility might have removed the row in SYSIBM.SYSCOPY. If the row was removed and the image copy is a full image copy with SHRLEVEL REFERENCE, use DSN1COPY to restore the table space or data set.

**Restriction:** If you use DSN1COPY for point-in-time recovery, the table space is not recoverable with the RECOVER utility. Because DSN1COPY runs outside of the control of Db2, Db2 is not aware that you recovered to a point in time. If possible, use DSN1COPY to recover the affected table space after a point-in-time recovery. Then perform the following steps:

1. Remove old image copies by using the MODIFY RECOVERY utility with the AGE option.
2. Create one or more full image copies by using the COPY utility with the SHRLEVEL REFERENCE option.

**Procedure**

To restore table spaces with DSN1COPY:

1. Delete data in any excess partitions from the table space before you apply the DSN1COPY utility.

   You can use the DSN1COPY utility to restore a partition or an entire table space for a partition-by-growth table space. The total number of partitions in the DSN1COPY input data set might not be consistent with the number of partitions that are defined on the current table space. To avoid residual data, delete data in the excess partitions from the table space before you apply the DSN1COPY utility.

2. If the table space is organized by hash, ensure that the following values are the same in the source and target tables spaces:
   - SYSTABLESPACE.HASHDATAPAGES (IF PBG)
   - SYSTABLEPART.HASHDATAPAGES (IF PBR-UTS)

3. If you are restoring an XML table space by using a data set that was generated by DSN1COPY before DB2 10 new-function mode, complete the following steps before you run DSN1COPY:
   a. Alter the target XML table space so that the SEGSIZE value matches the SEGSIZE value of the original XML table space. Use the following statement:
      
      ```sql
      ALTER TABLESPACE mytablespace SEGSIZE 4
      ```
      
   b. Run the REORG TABLESPACE utility on the target XML table space.

4. Run the DSN1COPY utility. Make sure that you provide the correct sequence of image copies to DSN1COPY.

   DSN1COPY can restore the object to an incremental image copy, but it must first restore the previous full image copy and any intermediate incremental
image copies. These actions ensure data integrity. You are responsible for providing the correct sequence of image copies. Db2 cannot ensure the appropriate sequence.

5. Ensure that the associated indexes are also rebuilt or restored. This action applies to all user-defined indexes and all indexes that are generated by Db2. For example, this action applies to the document ID index of a table with XML columns or the overflow index of a hash access table space.

Related reference:

- “Syntax and options of the DSN1COPY control statement” on page 964
- Chapter 21, “MODIFY RECOVERY,” on page 387
- “Syntax and options of the COPY control statement” on page 135
- “Syntax and options of the REORG TABLESPACE control statement” on page 556
- SYSIBM.SYSCOPY table (Db2 SQL)
- SYSIBM.SYSTABLESPACE table (Db2 SQL)
- SYSIBM.SYSTABLEPART table (Db2 SQL)
- ALTER TABLESPACE (Db2 SQL)

Related information:

- DSNU519I (Db2 Messages)

---

**Printing with DSN1COPY**

If you want to print one or more pages without invoking the utility’s copy function, use DSN1PRNT to avoid unnecessary reading of the input file.

**About this task**

When you use DSN1COPY for printing, you must specify the PRINT parameter. The requested operation takes place only for the specified data set. If the input data set belongs to a nonpartitioned table space or index space that is larger than 2 GB, specify the correct data set. Alternatively, if it is a partitioned table space or partitioned index, specify the correct data set. For example, DSN1COPY prints a page range in the second partition of a four-partition table space. DSN1COPY does this by specifying NUMPARTS(4) and the data set name of the second data set in the VSAM group (DSN=....A002).

To print a full image copy data set (rather than recovering a table space), specify a DUMMY SYSUT2 DD statement, and specify the FULLCOPY parameter.

---

**Copying tables from one subsystem to another**

You can copy tables from one subsystem to another by using the DSN1COPY utility. When you copy these tables, ensure that the object metadata on the target subsystem matches the object metadata on the source subsystem. Object metadata includes items such as the number of columns, column type, table space type, and version information.

**About this task**

**Recommendation:** Do not use DSN1COPY to copy XML table spaces from one subsystem to another. Documents in XML table spaces have dependencies on Db2 catalog tables and on tables in the XML schema repository database (DSNXSR). In
particular, XML documents in XML table spaces have unique strings IDs that must match values in catalog table SYSIBM.SYSXMLSTRINGS. Documents might also have XSR object IDs that must match values in XML schema repository table SYSIBM.XSROBJECTS. If you copy XML table spaces to from one subsystem to another, the string IDs and XSR object IDs in the XML documents will not match the values in SYSIBM.SYSXMLSTRINGS or SYSIBM.XSROBJECTS on the target subsystem.

**Procedure**

To copy a table space and its tables from one subsystem to another:

1. Check for and insert missing system pages into the table space on the source system.
   
   **Important:** This step makes the tables in the table space self-describing. When a table space is self-describing, Db2 does not need to search the catalog or directory for table format information, which lessens the possibility of errors after a table is copied to the target subsystem.
   
   a. Run REPAIR CATALOG TEST to check for missing system pages for tables.
      
      If a table space has any missing system pages, message DSNU667I is issued, with this additional information: MISSING SYSTEM PAGE IN THE PAGE SET.
   
   b. If the table space has any missing system pages for tables that are not in version 0 format, and no SQL operations or utilities have been run that store the current table version in the table space, skip to step 2.
   
   c. If the table space has any missing system pages for tables in the table space that are in version 0 format, run REPAIR INSERTVERSIONPAGES SETCURRENTVERSION. to insert system pages into the table space that describe those tables. Tables that are in version 0 format have had no version-changing alter operations.
      
      REPAIR INSERTVERSIONPAGES SETCURRENTVERSION performs the following actions:
      
      * Inserts missing system pages for tables that are in version 0 format into the table space.
      * Updates the version numbers in the VERSION column of the SYSIBM.SYSTABLES catalog table for all tables in the table space that are at version 0 to the same version number as in the CURRENT_VERSION column in SYSIBM.SYSTABLESPACE.
   
   d. Run REORG TABLESPACE on the table space on the source system to materialize all data rows to the current table space version, and to roll up all table space versions into the current table space version.

2. If a table space or table does not exist on the target subsystem, create it.

   If a table has an identity column, specify that column as follows:

   a. Issue a SELECT statement on the source subsystem to query the SYSIBM.SYSSEQUENCES entry that corresponds to the identity column for the table on the source subsystem.
   
   b. Add the INCREMENT value to the MAXASSIGNEDVAL value to determine the next value (nv) for the identity column.
   
   c. For the table on the target subsystem, specify nv for the START WITH value.
d. Make all of the attributes for the identity column on the target table the same as the identity column attributes for the source table.

3. If the table space or table exists on the target subsystem, examine the table space, table, and column definitions in the catalog to ensure that the definitions are the same on the source and target subsystems.

4. Query the DBID, PSID, and OBID of the object in the target subsystem. If the values are not the same as the source object, specify the DBID, PSID, and OBID as part of the OBIDXLAT data set for DSN1COPY.

5. Stop the table space on the source and target subsystems.

6. Run the DSN1COPY utility with the OBIDXLAT and RESET options. In the SYSXLAT data set, specify the proper mapping of table database object identifiers (OBIDs) for the table space from the source to the target subsystem.

7. Start the table space on the source and target subsystems for read/write access.

8. Run REPAIR CATALOG TEST on the table space on the target subsystem to ensure that the catalog information matches the page set information.
   Take one of the following actions:
   • If REPAIR CATALOG TEST ends with return code 0, continue to the next step.
   • If REPAIR CATALOG TEST ends with return code 4, run REPAIR CATALOG to fix any mismatches.
   • If REPAIR CATALOG TEST ends with return code 8, go to step 3 to begin the process of copying the data again.

REPAIR CATALOG updates the following columns:
   • OLDEST_VERSION in SYSTABLEPART
   • VERSION in SYSTABLES
   • OLDEST_VERSION and CURRENT_VERSION in SYSTABLESPACE

9. If there are any indexes that are defined on the table on the source subsystem, but are not defined on the table on the target subsystem, create those indexes.

10. Run REBUILD INDEX on all indexes that are defined on the table on the target subsystem.

Related reference:

“Syntax and options of the REPAIR control statement” on page 670
“Data sets that DSN1COPY uses” on page 972
“Syntax and options of the REBUILD INDEX control statement” on page 428

Sample DSN1COPY control statements

Use the sample control statements as models for developing your own DSN1COPY control statements.

If you run online REORG with FASTSWITCH behavior, the fifth-level qualifier in the data set name can be either I0001 or J0001. For clone tables, the data set can also be I0002 or J0002. These examples use I0001.
Example 1: Checking input data set before copying

The following statement specifies that the DSN1COPY utility is to copy the data set that is identified by the SYSUT1 DD statement to the data set that is identified by the SYSUT2 DD statement. Before DSN1COPY copies this data, the utility is to check the validity of the input data set.

```
//RUNCOPY EXEC PGM=DSN1COPY,PARM='CHECK'
//* COPY VSAM TO SEQUENTIAL AND CHECK PAGES
//STEPLIB DD DSN=PDS CONTAINING DSN1COPY
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=DSNCAT.DSNDBC.DSNDB01.SYSUTILX.I0001.A001,DISP=OLD
//SYSUT2 DD DSN=TAPE.DS,UNIT=TAPE,DISP=(NEW,KEEP),VOL=SER=UTLBAK
```

Example 2: Translating the Db2 internal identifiers

The statement in this example specifies that DSN1COPY is to copy the data set that is identified by the SYSUT1 DD statement to the data set that is identified by the SYSUT2 DD statement. The OBIDXLAT option specifies that DSN1COPY is to translate the OBIDs before the data set is copied. The OBIDs are provided as input on the SYSXLAT DD statement. Because the OBIDXLAT option is specified, DSN1COPY also checks the validity of the input data set, even though the CHECK option is not specified.

```
//EXECUTE EXEC PGM=DSN1COPY,PARM='OBIDXLAT'
//STEPLIB DD DSN=PDS CONTAINING DSN1COPY
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=DSNCAT.DSNDBC.DSNDB01.DSN8D11P.DSN8S11C.I0001.A001,DISP=OLD
//SYSUT2 DD DSN=DSNC910.DSNDBC.DSN8D11P.DSN8S11C.I0001.A001,DISP=OLD
//SYSXLAT DD *
260,280
2,10
3,55
6,56
7,57
/*
```

Figure 107. Example DSN1COPY statement with the OBIDXLAT option.

Example 3: Printing a single page of a partitioned table space

The following statement specifies that DSN1COPY is to print page 2002A1 of the table space in the data set that is identified by the SYSUT1 DD statement. This table space has eight partitions, as indicated by the NUMPARTS option.

```
//EXECUTE EXEC PGM=DSN1COPY,PARM='PRINT(2002A1),NUMPARTS(8)'
//SYSDUMP DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//SYSUT2 DD DUMMY
//SYSUT1 DD DSN=DSNCAT.DSNDBC.MMRDB.PARTEMP1.I0001.A003,DISP=OLD
```

Example 4: Printing 16 pages of a nonpartitioning index

The following statement specifies that DSN1COPY is to print 16 pages of a nonpartitioning index in the data set that is identified by the SYSUT1 DD statement. The pages range from page F0000 to page F000F, as indicated by the PRINT option. The maximum data set size is 64 MB, as indicated by the PIECESIZ option.
Example 5: Copying individual partitions of a partitioned table space

In the example in the following figure, the two job steps specify that DSN1COPY is to copy partitions 1501 and partition 1502 from image copy data sets into a partitioned table space. In the two SYSUT2 DD statements, the fifth-level qualifier in the data set name can differ, because each job step lists an individual partition. The FULLCOPY option is used in both steps to indicate that the input data set is a full image copy. The NUMPARTS option indicates that the input data set has 1600 partitions. The RESET option resets to 0 the high-formatted page number in the header page. Because this option is specified, DSN1COPY checks the validity of the input data, even though the CHECK option is not specified.

Example 6: Copying all partitions of a partitioned table space

The following statement specifies that DSN1COPY is to copy data into all partitions of a partitioned table space by using a full image copy of the table space as input. The input image copy has 16 partitions, as indicated by the NUMPARTS option. You must ensure that the fifth-level qualifier in the data set name is the same, either I0001 or J0001, for all partitions of the output table space before running this type of job stream.

Example 7: Using DSN1COPY with UTS table spaces

The following statements specify that DSN1COPY is to copy a UTS table space VSAM data set to a sequential data set.
Example 8: Specifying Unicode output for DSN1COPY

When you specify the UNICODE option for DSN1COPY, you are not going to see non-Latin Unicode characters, such as Japanese characters, in your output. When you specify the UNICODE option, DSN1COPY takes the hexadecimal data and formats it as ASCII instead of the default EBCDIC.

A problem might arise when the data that you want DSN1COPY to handle is in UTF-16. In the case of UTF-16 data, DSN1COPY takes only the second byte of the data and formats that part of the data as ASCII. Thus, the output might not be correct. For example, the UTF-16 hexadecimal values X'0030' and X'1130' are both output as 0, because the first byte of each ("00" and "11" respectively) is ignored. The remaining part ("30") is interpreted as an ASCII 0. In UTF-16, X'0030' is the hexadecimal value for 0, but X'1130' is the hexadecimal value for a Hangul character.

In the following DSN1COPY example, notice the three bold hexadecimal values: X'0041', X'0141', and X'0241'. The output for all three of these values is A.A.A, even though they each correspond to different characters in UTF-16. (X'0041' is A, X'0141' is ₩, and X'0241' is the Latin capital character for glottal stop.)
Contents of the input data set in ASCII:

*...u....3................*
*A.A.A. . . . . .............*
.... LINES ARE ALL ZERO.
*................................*

**Related information:**

[UTFs (Db2 Internationalization Guide)]

**Example 9: Defining output data sets for multi-piece nonpartitioned table spaces**

The following statements specify that DSN1COPY is to copy data into all pieces of a segmented table space by using a full image copy of another segmented table space as input.

1. Create the target segmented table space by specifying appropriate primary and secondary quantities.
2. Define data sets for all subsequent pieces.
   
   In the following example, the data set for the second piece is 'DSNCAT.DSNDBC.TESTDB.TS01.I0001.A002'. Use the MODEL option, which causes the new data set to be created like the first data set.

   ```
   //ALCVSAM EXEC PGM=IDCAMS
   //SYSPRINT DD SYSOUT=* 
   //SYSIN DD *
   DEFINE CLUSTER +
   (NAME('DSNCAT.DSNDBC.TESTDB.TS01.I0001.A002') +
   MODEL('DSNCAT.DSNDBC.TESTDB.TS01.I0001.A001') +
   DATA +
   (NAME('DSNCAT.DSNDBD.TESTDB.TS01.I0001.A002') +
   MODEL('DSNCAT.DSNDBD.TESTDB.TS01.I0001.A001')) )
   /*

   3. Run DSN1COPY.

   //DSN1COPY EXEC PGM=DSN1COPY, 
   // PARM='FULLCOPY' 
   //SYSDUMP DD SYSOUT=A 
   //SYSPRINT DD SYSOUT=A 
   //SYST1 DD DISP=SHR,DSN=PROD.IMAGE.COPY.DSNUMALL 
   //SYST2 DD DISP=OLD,DSN=DSNCAT.DSNDBD.TESTDB.TS01.I0001.A001
Chapter 45. DSN1LOGP

The DSN1LOGP stand-alone utility formats the contents of the recovery log for display.

The two recovery log report formats are:
- A detail report of individual log records. This information helps IBM Support personnel analyze the log in detail. (This information does not include a full description of the detail report.)
- A summary report, which helps you:
  - Perform a conditional restart
  - Resolve indoubt threads with a remote site
  - Detect problems with data propagation

You can specify the range of the log to process and select criteria within the range to limit the records in the detail report. For example, you can specify:
- One or more units of recovery that are identified by URID
- A single database

By specifying a URID and a database, you can display recovery log records that correspond to the use of one database by a single unit of recovery.

DSN1LOGP can print the log records for both base and clone table objects.

Environment

DSN1LOGP runs as a batch z/OS job.

DSN1LOGP runs on archive data sets, but not active data sets, when Db2 is running.

Authorization required

DSN1LOGP does not require authorization. However, if any of the data sets is RACF-protected, the authorization ID of the job must have RACF authority.

Required data sets

When you execute DSN1LOGP, provide the following data definition (DD) statements:

SYSPRINT
- DSN1LOGP writes all error messages, exception conditions, and the detail report to the SYSPRINT file. The logical record length (LRECL) is 131.

SYSIN
- DSN1LOGP specifies keywords in this file. The LRECL must be 80. Keywords and values must appear in characters 1 through 72. DSN1LOGP allows specification of as many as 50 control statements for a given job. DSN1LOGP concatenates all records into a single string.

SYSSUMRY
- DSN1LOGP writes the formatted output of a summary report to the SYSSUMRY file. The LRECL is 131.
DSN1LOGP identifies the recovery log by DD statements that are described in the stand-alone log services.

**Identifying log data sets**

You must identify to DSN1LOGP the log data sets that are to be processed by including at least one of the following DD statements.

**BSDS** The BSDS identifies and provides information about all active log data sets and archive log data sets that exist in your Db2 subsystem. When you identify the BSDS to DSN1LOGP, you must provide the beginning and ending RBAs for the range of the recovery log that you want displayed. DSN1LOGP then associates the beginning RBA specifications and the ending RBA specifications with the appropriate data set names.

**ACTIVE**

If the BSDS is not available, and if the active log data sets that are involved were copied and sent to you, use ACTIVE DD statements. Use one or more ACTIVE DD statements to specify the set of active log data sets that are to be processed by DSN1LOGP. If you used the REPRO command of Access Method Services for copying the active log, you must identify this data set in an ARCHIVE DD statement.

Each DD statement that you include identifies another active log data set. If you identify more than one active log data set, you must list the ACTIVE DD statements in ascending log RBA sequence. For example, ACTIVE1 must identify a portion of the log that is less than ACTIVE2, and ACTIVE2 must identify a portion of the log that is less than ACTIVE3. If you do not specify these DD statements correctly, errors that DSN1LOGP does not detect can occur. You can specify up to 16 of these active log data sets.

When you identify active log data sets, you do not need to use the RBASTART and RBAEND keywords (as you do when you identify the BSDS). DSN1LOGP scans all active log data sets that the job indicates only when the data sets are in the correct log RBA sequence.

**ARCHIVE**

If the BSDS is not available (as previously described under ACTIVE), you can specify which archive log data sets are to be processed by specifying one ARCHIVE DD statement, concatenated with one or more DD statements.

Each DD statement that you include identifies another archive log data set. If you identify more than one archive log data set, you must list the DD statements that correspond to the multiple archive log data sets in ascending log RBA sequence. If you do not specify this correctly, errors that DSN1LOGP does not detect can occur.

When you identify archive log data sets, you do not need to use the RBASTART and RBAEND keywords. DSN1LOGP scans all archive log data sets that are indicated by the job only when the data sets are in the correct log RBA sequence.

**Data sharing requirements**

When selecting log records from more than one Db2 subsystem, you must use all or one of the following DD statements to locate the log data sets:

GROUP
If you use GROUP or MxxBSDSs to locate the log data sets, you must use LRSNSTART to define the selection range.

Related concepts:
- [Obtaining Db2 Diagnosis Guide and Reference ()](#)

Related tasks:
- [Reading log records with OPEN, GET, and CLOSE (Db2 Administration Guide)](#)

## Syntax and options of the DSN1LOGP control statement

The DSN1LOGP utility control statement, with its multiple options, defines the function that the utility job performs.

### DSN1LOGP syntax diagram

```
value/offset statement:
```

[DSN1LOGP syntax diagram](#)
**Option descriptions**

To run DSN1LOGP, construct a batch job. Include the utility name, DSN1LOGP, on the EXEC statement.

Specify keywords in up to 50 control statements in the SYSIN file. Each control statement can have up to 72 characters. To specify no keywords, either use a SYSIN file with no keywords following it, or omit the SYSIN file from the job JCL.

If you specify more than one keyword, separate them by commas. You can specify the keywords in any order. You can include blanks between keywords, and also between the keywords and the corresponding values.

**RBASTART**(hex-constant)

Specifies the hexadecimal log RBA from which to begin reading. If the value does not match the beginning RBA of one of the log records, DSN1LOGP begins reading at the beginning RBA of the next record. Specify this keyword only once in the job. Alternative spellings: STARTRBA, ST.

*hex-constant* is a hexadecimal value that consists of 1 - 12 characters (6 bytes) or 1 - 20 characters (10 bytes) if the BSDS was converted by using the DSNJCNVT conversion program. Leading zeros are not required.

The default value is 0.

Db2 issues a warning if the value is not within the range of log records that is covered by the input log record information.

**RBAEND**(hex-constant)

Specifies the last valid hexadecimal log RBA to extract. If the specified RBA is in the middle of a log record, DSN1LOGP continues reading the log in an attempt to return a complete log record.

To read to the last valid RBA in the log up to the point of the 6 byte maximum, specify RBAEND(FFFFFFFFFFFF). To read to the last valid RBA in the log, specify RBAEND(FFFFFFFFFFFFFFFFFFFF). Specify this keyword only once in the job. Alternative spellings: ENDRBA, EN.

*hex-constant* is a hexadecimal value that consists of 1 - 12 characters (6 bytes) or 1 - 20 characters (10 bytes) if the BSDS was converted by using the DSNJCNVT conversion program. Leading zeros are not required.

The default value is FFFFFFFFFFFFFFFFFF.

Db2 issues a warning if the value is not within the range of log records that is covered by the input log record information.

RBAEND can be specified only if RBASTART is specified.

**LRSNSTART**(hex-constant)

Specifies the log record sequence number (LRSN) from which to begin the log scan. DSN1LOGP starts its processing on the first log record that contains an LRSN value that is greater than or equal to the LRSN value that is specified on LRSNSTART. The default LRSN is the LRSN at the beginning of the data set. Alternative spellings: STARTLRSN, STRTLRSN, and LRSNSTRT.
Specify this keyword only once in the job.

You must specify this keyword to search the member BSDSs and to locate the log data sets from more than one Db2 subsystem. You can specify either the LRSNSTART keyword or the RBASTART keyword to search the BSDS of a single Db2 subsystem and to locate the log data sets.

If you specify both LRSNSTART and LRSNEND, values greater than 12 characters must be the same length.

Db2 issues a warning if the value is not within the range of log records that is covered by the input log record information.

**LRSNEND**(hex-constant)

Specifies the LRSN value of the last log record that is to be scanned if LRSNSTART is also specified. If LRSNEND is not specified, the LRSNEND value is either current end of the log (X’FFFFFFFFFFFFFFFFFFFF’) or the LRSN value for the end of the data set.

DSN1LOGP ends its processing on the last log record that contains an LRSN value that is greater than or equal to the LRSN value that is specified on LRSNEND.

An alternative spelling for LRSNEND is ENDLRSN.

Specify this keyword only once in the job.

If you specify both LRSNSTART and LRSNEND, values greater than 12 characters must be the same length.

Db2 issues a warning if the value is not within the range of log records that is covered by the input log record information.

**DATAONLY**

Limits the log records in the detail report to those records that represent data changes (such as insert, page repair, and update space map).

The default value is DATAONLY(NO).

**(YES)**

Extracts log records for data changes only. For example, DATAONLY(YES), together with a DBID and OBID, reads only the log records that modified data for that DBID and OBID.

**(NO)**

Extracts all record types.

**SYSCOPY**

Limits the detail report to SYSCOPY log records.

The default value is SYSCOPY(NO).

**(YES)**

Includes only SYSCOPY log records in the detail report.

**(NO)**

Does not limit records to SYSCOPY records only.

**DBID**(hex-constant)

Specifies a hexadecimal database identifier (DBID). DSN1LOGP extracts only the records that are associated with that DBID. Specify this keyword only once in the job.

*hex-constant* is a hexadecimal value that consists of one to 4 characters. Leading zeros are not required.
You can find the DBID in any of the following ways:

- The DBID is displayed in many Db2 messages.
- You can find the DBID in the Db2 catalog for a specific object (for example, in the column that is named DBID of the SYSIBM.SYSTABLESPACE catalog table).
- When you select a DBID from a catalog table, the value is displayed in decimal format. Use the SQL HEX function in a SELECT statement to convert a DBID to hexadecimal format. The following SQL statements show this use of the HEX function:

```
SELECT NAME, DBNAME, HEX(DBID), HEX(PSID)
FROM SYSIBM.SYSTABLESPACE
WHERE NAME = 'table space name'
```

```
SELECT NAME, DBNAME, HEX(DBID), HEX(ISOBID)
FROM SYSIBM.SYSINDEXES
WHERE NAME = 'index name'
```

- You can use the DSN1PRNT utility to format the data sets for tables or indexes, and find the DBID in first 2 bytes of HPGOBID.

**OBID(hex-constant)**

Specifies a hexadecimal database object identifier, either a data page set identifier (PSID) or an index page set identifier (ISOBID). DSN1LOGP extracts only the records that are associated with that identifier.

*hex-constant* is a hexadecimal value that consists of one to 4 characters. Leading zeros are not required.

Whenever Db2 changes the data, the log record that describes the change identifies the database by DBID and the table space by page set ID (PSID). You can find the PSID column in the SYSIBM.SYSTABLESPACE catalog table.

You can also find a column that is named OBID in the SYSIBM.SYSTABLESPACE catalog table. That column contains the OBID of a file descriptor; do not confuse this value with the PSID, which is the information that you must include when you run DSN1LOGP.

Whenever Db2 changes an index, the log record that describes the change identifies the database (by DBID) and the index space (by index space OBID or ISOBID). You can find the ISOBID for an index space in the column that is named ISOBID in the SYSIBM.SYSINDEXES catalog table.

You can also find a column that is named OBID in the SYSIBM.SYSINDEXES catalog table. This column contains the identifier of a fan set descriptor; do not confuse this value with the ISOBID, which is the information that you must include when you run DSN1LOGP.

When you select either the PSID or the ISOBID from a catalog table, the value is displayed in decimal format. Use the SQL HEX function in your select statement to convert them to hexadecimal.

Alternatively, you can use the DSN1PRNT utility to format the data sets for tables or indexes, and find the PSID or ISOBID in the last 2 bytes of HPGOBID.

You can specify the OBID keyword up to 10 times for a single DSN1LOGP job. If you specify OBID, you must also specify DBID.

**PAGE(hex-constant)**

Specifies a hexadecimal page number. When data or an index is changed, a recovery log record is written to the log, identifying the object identifier and the page number of the changed data page or index page. Specifying a page...
number limits the search to a single page; otherwise, all pages for the combination of DBID and OBID are extracted. The log output also contains page set control log records for the specified DBID and OBID, and system event log records, unless DATAONLY(YES) is also specified.

*hex-constant* is a hexadecimal value that consists of a maximum of 8 characters.

You can specify a maximum of 100 PAGE keywords in a single DSN1LOGP job. You must also specify the DBID and OBID keywords that correspond to those pages.

The PAGE and RID keywords are mutually exclusive.

**RID(hex-constant)**

Specifies a record identifier, which is a hexadecimal value that consists of 10 characters, with the first 8 characters representing the page number and the last 2 characters representing the page ID map entry number. The option limits the log records that are extracted to those records that are associated with that particular record. The log records that are extracted include the following records:

- Those records that are directly associated with the RID, such as insert and delete
- The control records that are associated with the DBID and OBID specifications, such as page set open, page set close, set write, reset write, page set write, data set open, and data set close

You can specify a maximum of 40 RID keywords in a single DSN1LOGP job. You must also specify the DBID and OBID keywords that correspond to the specified records.

The PAGE and RID keywords are mutually exclusive.

**URID(hex-constant)**

Specifies a hexadecimal unit of recovery identifier (URID). Changes to data and indexes occur in the context of a Db2 unit of recovery, which is identified on the log by a BEGIN UR record. In the summary DSN1LOGP report, the URID is listed in the STARTRBA field in message DSN1162I. In the detail DSN1LOGP report, look for the subtype of BEGIN UR; the URID is listed in the URID field. Using the log RBA of that record as the URID value limits the extraction of information from the Db2 log to that unit of recovery.

*hex-constant* is a hexadecimal value that consists of 1 - 12 characters (6 bytes) or 1 - 20 characters (10 bytes) if the BSDS was converted by using the DSNJCNVT conversion program. Leading zeros are not required.

You can specify a maximum of 10 URID keywords in a single DSN1LOGP job.

**LUWID(luwid)**

Specifies up to 10 LUWIDs that DSN1LOGP is to include information about in the summary report.

*luwid* consists of three parts: an LU network name, an LUW instance number, and a commit sequence number. If you supply the first two parts, the summary report includes an entry for each commit that is performed in the logical unit of work (within the search range). If you supply all three parts, the summary report includes an entry for only that LUWID.

The LU network name consists of a one- to eight-character network ID, a period, and a one- to eight-character network LU name. The LUW instance.
number consists of a period, followed by 12 hexadecimal characters. The last
element of the LUWID is the commit sequence number of 4 hexadecimal
characters, preceded by a period.

**TYPE(hex-constant)**

Limits the log records that are extracted to records of a specified type. The
TYPE and SUBTYPE options are mutually exclusive.

If you specify TYPE, DSN1LOGP ignores the values that you specify for the
other keywords except for RBASTART, RBAEND, LRSNSTART, and
LRSNEND.

`hex-constant` indicates the type, as follows:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Page set control record</td>
</tr>
<tr>
<td>4</td>
<td>SYSCOPY utility record</td>
</tr>
<tr>
<td>10</td>
<td>System event record</td>
</tr>
<tr>
<td>20</td>
<td>UR control record</td>
</tr>
<tr>
<td>100</td>
<td>Checkpoint record</td>
</tr>
<tr>
<td>200</td>
<td>UR-UNDO record</td>
</tr>
<tr>
<td>400</td>
<td>UR-REDO record</td>
</tr>
<tr>
<td>800</td>
<td>Archive quiesce record</td>
</tr>
<tr>
<td>1000 - 8000</td>
<td>Assigned by the resource manager</td>
</tr>
</tbody>
</table>

**SUBTYPE(hex-constant)**

Restricts formatting to a particular subtype of unit of recovery undo and redo
log records (types 200 and 400). The TYPE and SUBTYPE options are mutually
exclusive.

`hex-constant` indicates the subtype, as follows:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Update data page</td>
</tr>
<tr>
<td>2</td>
<td>Format page or update space map</td>
</tr>
<tr>
<td>3</td>
<td>Update space map bits</td>
</tr>
<tr>
<td>4</td>
<td>Update to index space map</td>
</tr>
<tr>
<td>5</td>
<td>Update to index page</td>
</tr>
<tr>
<td>6</td>
<td>DBA table update log record</td>
</tr>
<tr>
<td>7</td>
<td>Checkpoint DBA table log record</td>
</tr>
<tr>
<td>9</td>
<td>DBD virtual memory copy</td>
</tr>
<tr>
<td>A</td>
<td>Exclusive lock on page set partition or DBD</td>
</tr>
<tr>
<td>B</td>
<td>Format file page set</td>
</tr>
<tr>
<td>C</td>
<td>Format index page set</td>
</tr>
<tr>
<td>F</td>
<td>Update by repair (first half if 32 KB)</td>
</tr>
</tbody>
</table>
10  Update by repair (second half if 32 KB)
11  Allocate or deallocate a segment entry
12  Undo/redo log record for modified page or redo log record for formatted page
14  Savepoint
15  Other Db2 component log records that are written for RMID 14
17  Checkpoint record of modified page set
19  Type 2 index update
1A  Type 2 index undo/redo or redo log record
1B  Type 2 index change notification log record
1C  Type 2 index space map update
1D  DBET log record with exception data
1E  DBET log record with LPL/GRECP data
65  Change Data Capture diagnostic log
81  Index dummy compensation log record
82  START DATABASE ACCESS (FORCE) log record

The VALUE and OFFSET options must be used together. You can specify a maximum of 10 VALUE-OFFSET pairs. The SUBTYPE parameter is required when you specify the VALUE and OFFSET options.

**VALUE**(*hex-constant*)

Specifies a value that must appear in a log record that is to be extracted.

*hex-constant* is a hexadecimal value that consists of a maximum of 64 characters and must be an even number of characters.

The SUBTYPE keyword must be specified before the VALUE option.

**OFFSET**(*hex-constant*)

Specifies an offset from the log record header at which the value that is specified in the VALUE option must appear.

*hex-constant* is a hexadecimal value that consists of a maximum of 8 characters.

If you specify the OFFSET option, you must specify the SUBTYPE keyword.

**SUMMARY**

Summarizes all recovery information within the RBASTART and RBAEND specifications. You can use summary information to determine what work is incomplete when Db2 starts. You cannot limit the output of the summary report with any of the other options, except by using the FILTER option with a URID or LUWID specification.

The default value is **SUMMARY(NO)**.

**YES**

Generates both a detail and summary report.

**NO**

Generates only a detail report.
Generates only a summary report.

FILTER
Restricts the summary report to include messages for only the specified URIDs and LUWIDs. Specify this option only once.

The SUMMARY keyword must be specified before FILTER.

CHECK(DATA)
Specifies that DSN1LOGP is to check the specified range of data pages for page regression. Any page regression errors are displayed in the detail and summary reports.

Related concepts:
“DSN1LOGP output” on page 1002
Related reference:
“Sample DSN1LOGP control statements” on page 999

Determining the PSID for base and clone objects

You can determine the PSID for base and clone objects by querying the SYSIBM.SYSTABLESPACE catalog table. You can specify the PSID on the DBID and OBID keywords of the DSN1LOGP utility control statement.

Procedure

To determine the PSID to specify for base or clone objects:

1. Determine the PSID by querying the SYSIBM.SYSTABLESPACE catalog table. The value is displayed in decimal format. Use the SQL HEX function in your select statement to convert the value to hexadecimal.

2. Determine the instance number of the clone or base object. You can determine the instance number in two ways:
   • Look at the TYPE column in the DISPLAY DATABASE command output. The output indicates the base and clone objects with a ‘B’ or a ‘C’ character respectively along with the data set instance number.
   • Look at the Db2 catalog. The SYSIBM.SYSTABLESPACE catalog table INSTANCE column indicates the current instance number of the base table.

3. Determine whether to alter the PSID value or leave the PSID value the same. For example, if the PSID value of the base or clone is '0009'X and the instance number is 1, specify a PSID value of '0009'X to DSN1LOGP. If the PSID of the base or clone is '0009'X and the instance number is 2, specify a PSID value of '8009'X to DSN1LOGP.

Related reference:
- "DISPLAY DATABASE (Db2) (Db2 Commands)"
- "SYSIBM.SYSTABLESPACE table (Db2 SQL)"

Archive log data sets on tape

If you store your archive logs on tape, the offload task constructs two files on tape during the archiving process. The first file is the BSDS, and the second file is a dump of the active log that the offload task is currently archiving.

If a failure occurs during the time that the offload task is archiving the BSDS, Db2 might omit the BSDS. In this case, the first file contains the active log.
If you perform archiving on tape, the first letter of the lowest-level qualifier varies for both the first and second data sets. The first letter of the first data set is B (for BSDS), and the first letter of the second data set is A (for archive). Hence, the archive log data set names all end in Axxxxxx, and the DD statement identifies each of them as the second data set on the corresponding tape:

\[ \text{LABEL=(Z,SL)} \]

When reading archive log data sets on tape (or copies of active log data sets on tape), add one or more of the following Job Entry Subsystem (JES) statements:

For the JES3 environment:

**JES3 environment JCL**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*<em>//<em>MAIN SETUP=JOB</em></em></td>
</tr>
<tr>
<td>Alert the z/OS operator to mount the initial volumes before the job executes.</td>
</tr>
<tr>
<td>*<em>//<em>MAIN HOLD=YES</em></em></td>
</tr>
<tr>
<td>Place the job in HOLD status until the operator is ready to release the job.</td>
</tr>
<tr>
<td><strong>TYPRUN=HOLD</strong></td>
</tr>
<tr>
<td>Perform the same function as *<em>//<em>MAIN HOLD=YES</em></em>. The system places the JCL on the JOB statement.</td>
</tr>
</tbody>
</table>

For the JES2 environment:

**JES2 environment JCL**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*<em>/<em>SETUP</em></em></td>
</tr>
<tr>
<td>Alert the z/OS operator to prepare to mount a specified list of tapes.</td>
</tr>
<tr>
<td>*<em>/<em>HOLD</em></em></td>
</tr>
<tr>
<td>Place the job in HOLD status until the operator has located the tapes and is ready to release the job.</td>
</tr>
<tr>
<td><strong>TYPRUN=HOLD</strong></td>
</tr>
<tr>
<td>Perform the same function as *<em>/<em>HOLD</em></em>. The system places the JCL on the JOB statement.</td>
</tr>
</tbody>
</table>

Alternatively, submit the job to a z/OS initiator that your operations center has established for exclusive use by jobs that require tape mounts. Specify the initiator class by using the CLASS parameter on the JOB statement, in both JES2 and JES3 environments.

**Related reference:**

[MVS JCL Reference](#)

---

**Sample DSN1LOGP control statements**

Use the sample control statements as models for developing your own DSN1LOGP control statements.

**Example 1: Extracting information from the recovery log with an available BSDS**

The following example shows how to extract information from the recovery log when you have the BSDS available. The extraction starts at the log RBA of
The DSN1LOGP utility identifies the table or index space by the DBID of X'10A' (266 decimal) and the OBID of X'1F' (31 decimal).

/*
You can think of the Db2 recovery log as a large sequential file. When recovery log records are written, they are written to the end of the log. A log RBA is the address of a byte on the log. Because the recovery log is larger than a single data set, the recovery log is physically stored on many data sets. Db2 records the RBA ranges and their corresponding data sets in the BSDS. To determine which data set contains a specific RBA, read the information about the DSNJU004 utility. During normal Db2 operation, messages are issued that include information about log RBAs.

Example 2: Extracting information from the active log when the BSDS is not available

The following example shows how to extract the information from the active log when the BSDS is not available. The extraction includes log records that apply to the table space or index space that is identified by the DBID of X'10A' and the OBID of X'1F'. The only information that is extracted is information that relates to page numbers X'3B' and X'8C', as identified by the PAGE options. You can omit beginning and ending RBA values for ACTIVE\* or ARCHIVE DD statements because the DSN1LOGP search includes all specified ACTIVE\* DD statements. The DD statements ACTIVE1, ACTIVE2, and ACTIVE3 specify the log data sets in ascending log RBA range. Use the DSNJU004 utility to determine what the log RBA range is for each active log data set. If the BSDS is not available and you cannot determine the ascending log RBA order of the data sets, you must run each log data set individually.

/*
Example 3: Extracting information from the archive log when the BSDS is not available

The example in the following figure shows how to extract the information from archive logs when the BSDS is not available. The extraction includes log records that apply to a single unit of recovery (whose URID is X'61F321'). Because the BEGIN UR is the first record for the unit of recovery and is at X'61F321', the beginning RBA is specified to indicate that it is the first RBA in the range from which to extract recovery log records. Also, because no ending RBA value is specified, all specified archive logs are scanned for qualifying log records. The
specification of DBID(4) limits the scan to changes that the specified unit of recovery made to all table spaces and index spaces in the database whose DBID is X'4'.

Example 4: Use DSN1LOGP with the SUMMARY option

The DSN1LOGP SUMMARY option allows you to scan the recovery log to determine what work is incomplete at restart time. You can specify this option either by itself or when you use DSN1LOGP to produce a detail report of log data. Summary log results appear in SYSSUMRY; therefore, you must include a SYSSUMRY DD statement as part of the JCL with which you execute DSN1LOGP.

The following example produces both a detail and a summary report that uses the BSDS to identify the log data sets. The summary report summarizes all recovery log information within the RBASTART and RBAEND specifications. You cannot limit the output of the summary report with any of the other options, except by using the FILTER option with a URID or LUWID specification. RBASTART and RBAEND specification use depends on whether a BSDS is used.

This example is similar to Example 1, in that it shows how to extract the information from the recovery log when you have the BSDS available. However, this example also shows you how to specify a summary report of all logged information between the log RBA of X'AF000' and the log RBA of X'B3000'. This summary is generated with a detail report, but it is printed to SYSSUMRY separately.

Example 5: Use DSN1LOGP on all members of a data sharing group

The following example shows how to extract log information that pertains to the table space that is identified by DBID X'112' and OBID X'1D' from all members of a data sharing group.
Example 6: Use DSN1LOGP on a single member of a data sharing group

The following example shows how to extract log information that pertains to the table space that is identified by DBID X'112' and OBID X'1D' from a single member of a data sharing group.

```plaintext
//STEP1 EXEC PGM=DSN1LOGP
//STEPLIB DD DSN=PDS containing DSN1LOGP
//SYSPRINT SYSOUT=A
//SYSABEND SYSOUT=A
//GROUP DD DSN=DSNDB0G.BSDS01,DISP=SHR
//SYSIN DD *
   DATAONLY (YES)
   LRSNSTART (00CA21F57927B1D48000)
   LRSNEND  (00CA21F57927B2BB000)
   DBID (112) OBID(1D)
/*
```

Example 5: Use DSN1LOGP on all members of a data sharing group

The following example shows how to extract log information that pertains all log records matching DBID X'112' and OBID X'1D' from the data sharing group after the LRSN X'00CAFFFFFFFFF1D48000'.

```plaintext
//STEP1 EXEC PGM=DSN1LOGP
//STEPLIB DD DSN=PDS containing DSN1LOGP
//SYSPRINT SYSOUT=A
//SYSABEND SYSOUT=A
//GROUP DD DSN=DSNDB0G.BSDS01,DISP=SHR
//SYSIN DD *
   DATAONLY (YES)
   LRSNSTART (A7951A001AD5) LRSNEND (A7951A003B6A)
   DBID (112) OBID(1D)
/*
```

Related reference:
Chapter 41, “DSNJU004 (print log map),” on page 927

**DSN1LOGP output**

One intended use of this utility is to aid in determining and correcting system problems. When diagnosing Db2 problems, you might need to refer to licensed documentation to interpret output from this utility.

**Reviewing DSN1LOGP output**

With the SUMMARY option, you can produce a summary report, a detail report, or both. You can also use the CHECK(DATA) option to produce a summary and detail report of page regression errors.
For data sharing, you might see multiple log records with the same LRSN value on a single Db2 data sharing member.

**Description of the summary report**

The summary report in the following figure contains a summary of completed events, consisting of an entry for each completed unit of work. Each entry shows, among other information, the start time, user, and all page sets that were modified. When possible, the report shows whether an object is LOGGED or NOT LOGGED.

The summary report is divided into two distinct sections:

- The first section is headed by the following message:
  
  `DSN1150I SUMMARY OF COMPLETED EVENTS`

- The second section is headed by the following message:

  `DSN1157I RESTART SUMMARY`

The first section lists all completed units of recovery (URs) and checkpoints within the range of the log that is scanned. Events are listed chronologically, with URs listed according to when they were completed and checkpoints listed according to when the end of the checkpoint was processed. The page sets that are changed by each completed UR are listed. If a log record that is associated with a UR is unavailable, the attribute INFO=PARTIAL is displayed for the UR. Otherwise, the UR is marked INFO=COMPLETE. A log record that is associated with a UR is unavailable if the range of the scanned log is not large enough to contain all records for a given UR.

The DISP attribute can be one of the following values: COMMITTED, ABORTED, INFLIGHT, IN-COMMIT, IN-ABORT, POSTPONED ABORT, or INDOUBT. The DISP attributes COMMITTED and ABORTED are used in the first section; the remaining attributes are used in the second section.

The list in the second section shows the work that is required of Db2 at restart as it is recorded in the log that you specified. If the log is available, the checkpoint that is to be used is identified, as is each outstanding UR, together with the page sets it changed. Each page set with pending writes is also identified, as is the earliest log record that is required to complete those writes. If a log record that is associated with a UR is unavailable, the attribute INFO=PARTIAL is displayed, and the identification of modified page sets is incomplete for that UR.

```
<table>
<thead>
<tr>
<th>DSN1212I DSN1LOGD FIRST LOG LRSN ENCOUNTERED 00CA21F57927B1D48000</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSN1150I SUMMARY OF COMPLETED EVENTS</td>
</tr>
<tr>
<td>START DATE=00.161 TIME=11:27:30 DISP=COMMITTED INFO=COMPLETE</td>
</tr>
<tr>
<td>STARTRBA=000000000002B836475 ENDRBA=000000000002B837024</td>
</tr>
<tr>
<td>STARTLRSN=00CA21F57955B9AE000 ENDLRSN=00CA21F57953A83EC000</td>
</tr>
<tr>
<td>NID=** LUWID=DSNCAT.SYEC1DB2.CA21F5792E8A.0001</td>
</tr>
<tr>
<td>COORDINATOR=** PARTICIPANTS=**</td>
</tr>
<tr>
<td>DATABASE=0001=DSNDB01 PAGE SET=00CF=SYSLGRNX</td>
</tr>
<tr>
<td>DATABASE=0001=DSNDB01 PAGE SET=0087=DSNLLX01</td>
</tr>
<tr>
<td>DATABASE=0001=DSNDB01 PAGE SET=0086=DSNLLX02</td>
</tr>
<tr>
<td>DSN1150I SUMMARY OF COMPLETED EVENTS</td>
</tr>
<tr>
<td>START DATE=00.161 TIME=11:27:30 DISP=COMMITTED INFO=COMPLETE</td>
</tr>
<tr>
<td>STARTRBA=000000000002B8374EF ENDRBA=000000000002B837CB1</td>
</tr>
<tr>
<td>STARTLRSN=00CA21F57956411DE000 ENDLRSN=00CA21F57959841E68000</td>
</tr>
<tr>
<td>NID=** LUWID=DSNCAT.SYEC1DB2.CA21F5795571.0001</td>
</tr>
<tr>
<td>COORDINATOR=** PARTICIPANTS=**</td>
</tr>
<tr>
<td>DATABASE=0001=DSNDB01 PAGE SET=00CF=SYSLGRNX</td>
</tr>
</tbody>
</table>
```
Description of the detail report

The detail report in the following figure includes the following records:

- Redo and undo log records
- System events log records, including begin and end checkpoint records, begin current status rebuild records, and begin forward and backward recovery records
- Page set control log records, including open and close page set log records, open and close data set log records, set write, reset write, and page set write log records
- UR control log records for the complete or incomplete unit of recovery

You can reduce the volume of the detail log records by specifying one or more of the optional keywords in the DSN1LOGP utility control statement.
Description of data propagation information in the summary report

The sample output in the following figure shows information from the DSN1LOGP summary report about log records of changes to Db2 tables that were defined with DATA CAPTURE CHANGES.

The fields show the following information:

- **START RBA** and **END RBA** show the first and last RBAs that are captured for the unit of recovery that was not retrieved. The range that the start and end RBA encompass can include one or all of the SQL statements within the scope of the unit of recovery.
- **TABLE LIST OVERFLOW** indicates whether more than 10 distinct data capture table IDs were updated by this unit of recovery. This example indicates that no overflow occurred.
- **LR WRITTEN** shows the number of written log records that represented changes to tables that were defined for data capture and were available to the DB2CDCEX routine. Recursive SQL changes from DB2CDCEX and changes from other attachments that are not associated with DB2CDCEX are not included. If you receive a value of 2147483647, an overflow occurred and the count is not valid.
- **LR RETRIEVED** is the number of captured RBAs that were retrieved by DB2CDCEX. If you receive a value of 2147483647, an overflow occurred and the count is not valid.
- **LR NOT RETRIEVED** is the difference between the number of written log records (LR WRITTEN) and the number of retrieved log records (LR RETRIEVED). The following example output shows that four log records were written, and none were retrieved.

**Figure 110. Sample data propagation information from the summary report**

Description of the report on page regression errors

DSN1LOGP reports page regression errors when you specify the CHECK(DATA) option. The value of the SUMMARY option determines whether the utility creates a detail report, a summary report, or both.

A detail report contains the following information for each page regression error:
A summary report contains the total number of page regressions that the utility found as well as the following information for each table space in which it found page regression errors:

- Database name
- Table space name
- DBID
- OBID

If no page regression errors are found, DSN1LOGP outputs a single message that no page regression errors were found.

The sample output in the following figure shows the detail report when page regression errors are found.

![Sample Output](image)

**Command text in DSN1LOGP output**

Command text is logged automatically. When you run DSN1LOGP with TYPE(0010), the output includes records for commands. Command record output has TYPE(SYSTEM EVENT) and SUBTYPE(TRACE RECORD). The data is in the form of an IFCID 0090 trace record. For example, the following output from DSN1LOGP shows a -STOP DB2 command:

```
00006BF8999  LRSN(C6CD403E83A8)  TYPE(SYSTEM EVENT)
  SUBTYPE(TRACE RECORD)
  +LRH+ 01400004 00100041 10800000 00000000 00000000 00000000 00000000 0000C6CD 403E83A8 0000
  0000 011A0000 00000028 00FF2001 00000014 00130001 0008600E E36D740 E36D740 C4CF2F16 + 2 STOP DB2
  0020 81A82000 00000040 00560117 005A02A1 16180930 C4E2D5C1 C6CD403E C6CD403E B3920000 +a ! DSNAF k
  0040 00000006 00000006 00000000 00E2D3C5 C3F14040 40404040 40404040 40404040 40404040 + STLEC1 DSNA
  0060 40404040 E2E8C5C3 F1C4CF2 C6CD403E B3770001 00000000 00000000 00000000 00000000 00000000 + SYEC1D82F 3090
  0080 02002002 EB20D709 4040F0F2 F34B7C8C E2C3D5F6 00F2E5C1 01C14040 01C14040 40404040 + SYSOPR 023.GSCS602VA1A
  00A0 40404040 4040E2EB EB20D709 04044000 00000000 00000000 00000000 00000000 00000000 + SYSOPR
```
Interpreting error codes

When an error occurs, DSN1LOGP formats a reason code from the Db2 stand-alone log service in the SYSPRINT output.

DSN1LOGP can abnormally terminate with a user abend code of X'099'. DSN1LOGP finds the corresponding abend reason code in register 15 (at the time of error). If the specified RBA or LRSN range was not found in the input data sets DSN1LOGP will terminate with return code 4.

Related reference:

 Registers and return codes (Db2 Administration Guide)
Chapter 46. DSN1PRNT

With the DSN1PRNT stand-alone utility, you can print Db2 VSAM data sets. These data sets can contain table spaces or index spaces, image copy data sets, and sequential data sets that contain Db2 table spaces or index spaces.

A Db2 VSAM data set is a single piece of a nonpartitioned table space or index, a single partition of a partitioned table space or index, or a FlashCopy image copy data set. The input must be a single z/OS sequential or VSAM data set. Concatenation of input data sets is not supported.

Using DSN1PRNT, you can print hexadecimal dumps of Db2 data sets and databases. If you specify the FORMAT option, DSN1PRNT formats the data and indexes for any page that does not contain an error that would prevent formatting. If DSN1PRNT detects such an error, it prints an error message just before the page and dumps the page without formatting. Formatting resumes with the next page.

Compressed records (including the compressed data of dictionary pages) are printed in compressed format.

DSN1PRNT is especially useful when you want to identify the contents of a table space or index. You can run DSN1PRNT on image copy data sets and on table spaces and indexes. DSN1PRNT accepts an index image copy as input when you specify the FULLCOPY option.

You cannot run DSN1PRNT on concurrent copies.

DSN1PRNT is compatible with LOB table spaces, when you specify the LOB keyword and omit the INLCOPY keyword.

DSN1PRNT does not decrypt any encrypted data; the utility displays the data as is.

Output

One intended use of this utility is to aid in determining and correcting system problems. When diagnosing Db2, you might need to refer to licensed documentation to interpret output from this utility.

Environment

Run DSN1PRNT as a z/OS job.

You can run DSN1PRNT even when the Db2 subsystem is not operational. If you choose to use DSN1PRNT when the Db2 subsystem is operational, ensure that the Db2 data sets that are to be printed are not currently allocated to Db2.

To make sure that a data set is not currently allocated to Db2, issue the Db2 STOP DATABASE command, specifying the table spaces and indexes that you want to print.
Authorization required

No special authorization is required. However, if any of the data sets is RACF protected, the authorization ID of the job must have RACF authority.

Required data sets

DSN1PRNT uses the following DD statements:

**SYSPRINT**

Defines the data set that contains output messages from DSN1PRNT and all hexadecimal dump output.

**SYSUT1**

Defines the input data set. That data set can be a sequential data set or a VSAM data set.

Disposition for this data set must be specified as OLD (DISP=OLD) to ensure that it is not in use by Db2. Specify the disposition for this data set as SHR (DISP=SHR) only in circumstances where the Db2 STOP DATABASE command does not work.

The requested operation takes place only for the specified data set. If the input data set belongs to a nonpartitioned table space or index space that is larger than 2 GB, or if it is a partitioned table space or index space, you must ensure the correct data set is specified. For example, to print a page range in the second partition of a four-partition table space, specify NUMPARTS(4) and the data set name of the data set in the group of VSAM data sets comprising the table space. The following code shows the data set name:

DSN=...A002

If you run the online REORG utility with FASTSWITCH behavior, verify the data set name before running the DSN1PRNT utility. The fifth-level qualifier in the data set name alternates between I0001 and J0001 when using FASTSWITCH. If the table space has cloning, the fifth-level qualifier can be I0002 or J0002. Specify the correct fifth-level qualifier in the data set name to successfully execute the DSN1PRNT utility. To determine the correct fifth-level qualifier, query the IPREFIX column of SYSIBM.SYSTABLEPART for each data partition or the IPREFIX column of SYSIBM.SYINDEXPART for each index partition. If the object is not partitioned, use zero as the value for the PARTITION column in your query.

Related concepts:

[Obtaining Db2 Diagnosis Guide and Reference](#)

Syntax and options of the DSN1PRNT control statement

The DSN1PRNT utility control statement, with its multiple options, defines the function that the utility job performs.
Option descriptions

If you have the need to run DSN1PRNT with invocation parameters specify one or more of the following options on the EXEC statement.

Important: If you specify more than one parameter:
• Separate them by commas (no blanks).
• Specify them in any order.

Default settings for DSN1PRNT options are taken from the input data set header page. This default processing is recommended when running DSN1PRNT because incorrect parameter settings can result in unpredictable results.
When non-default user values are specified, DSN1PRNT compares the input data set header page settings against the user-specified values whenever possible. If a mismatch is detected, message DSN1930I is issued. The processing is performed with the user-specified values.

32K
Specifies that the SYSUT1 data set has a 32-KB page size. If you specify this option and the SYSUT1 data set does not have a 32-KB page size, DSN1COPY might produce unpredictable results.

PAGESIZE
Specifies the page size of the input data set that is defined by SYSUT1. Available page size values are 4K, 8K, 16K, or 32K. If you specify an incorrect page size, DSN1PRNT might produce unpredictable results.

If you do not specify the page size, DSN1PRNT tries to determine the page size from the input data set if the first page of the input data set is a header page. Db2 issues an error message if DSN1PRNT cannot determine the input page size. This might happen if the header page is not in the input data set, or if the page size field in the header page contains an invalid page size.

Related information:
“Determining the page size and data set size for DSN1PRNT” on page 1016

FULLCOPY
Specifies that a Db2 full image copy (not a DFSMSdss concurrent copy) of your data is to be used as input. If this data is partitioned, you also need to specify the NUMPARTS parameter to identify the number and length of the partitions.

If you specify FULLCOPY without including a NUMPARTS specification, DSN1PRNT assumes that the input file is not partitioned.

The FULLCOPY parameter must be specified when you use an image copy as input to DSN1PRNT. Omitting the parameter can cause error messages or unpredictable results.

Do not specify FULLCOPY if the input image copy is a FlashCopy image copy data set.

INCRCOPY
Specifies that an incremental image copy of the data is to be used as input. If the data is partitioned, also specify NUMPARTS to identify the number and length of the partitions. If you specify INCRCOPY without NUMPARTS, DSN1PRNT assumes that the input file is not partitioned.

The INCRCOPY parameter must be specified when you use an incremental image copy as input to DSN1PRNT. Omitting the parameter can cause error messages or unpredictable results.

INLCOPY
Specifies that the input data is to be an inline copy data set.

When DSN1PRNT is used to print a page or a page range from an inline copy that is produced by LOAD or REORG, DSN1PRNT prints all instances of the pages. The last instance of the printed page or pages is the last one that is created by the utility.

The INLCOPY parameter must be specified when an inline image copy is used as input to DSN1PRNT. Omitting the INLCOPY parameter can cause error messages or unpredictable results.

LARGE
Specifies that the input data set is a table space that was defined with the
The LARGE option, or an index on such a table space. If you specify LARGE, Db2 assumes that the data set has a 4-GB boundary. The recommended method of specifying a table space that was defined with the LARGE option is DSSIZE(4G).

If you omit the LARGE or DSSIZE(4G) option when it is needed, or if you specify LARGE for a table space that was not defined with the LARGE option, the results from DSN1PRNT are unpredictable.

If you specify LARGE, you cannot specify LOB or DSSIZE.

LOB
Specifies that the SYSUT1 data set is a LOB table space.
You cannot specify the INLCOPY option with the LOB parameter.
If you specify LOB, you cannot specify LARGE.
Db2 attempts to determine if the input data set is a LOB data set. If you specify the LOB option but the data set is not a LOB data set, or if you omit the LOB option but the data set is a LOB data set, Db2 issues an error message and DSN1PRNT terminates.

DSSIZE(integer G)
Specifies the data set size, in gigabytes, for the input data set. If you omit DSSIZE, Db2 obtains the data set size from the data set header page.
If you specify DSSIZE, integer must match the DSSIZE value that was specified when the table space was defined.

Related information:
“Determining the page size and data set size for DSN1PRNT” on page 1016

PIECESIZ(integer)
Specifies the maximum piece size (data set size) for nonpartitioned indexes.
The value that you specify must match the value that is specified when the secondary index was created or altered.
The defaults for PIECESIZ are 2G (2 GB) for indexes that are backed by non-large table spaces and 4G (4 GB) for indexes that are backed by table spaces that were defined with the LARGE option. This option is required if a print range is specified and the piece size is not one of the default values. If PIECESIZ is omitted and the index is backed by a table space that was defined with the LARGE option, the LARGE keyword is required for DSN1PRNT.
The subsequent keyword K, M, or G, indicates the units of the value that is specified in integer.
K Indicates that the integer value is to be multiplied by 1 KB to specify the maximum piece size in bytes. integer must be either 256 or 512.
M Indicates that the integer value is to be multiplied by 1 MB to specify the maximum piece size in bytes. integer must be a power of 2, between 1 and 512.
G Indicates that the integer value is to be multiplied by 1 GB to specify the maximum piece size in bytes. integer must be a power of two, between 1 and 256.

Valid values for piece size are:
• 1 MB or 1 GB
• 2 MB or 2 GB
• 4 MB or 4 GB
NUMPARTS(integer)
This parameter is not used if the target table space is a universal table space. DSSIZE is used instead.

PRINT(hexadecimal-constant,hexadecimal-constant)
Causes the SYSUT1 data set to be printed in hexadecimal format on the SYSPRINT data set. This option is the default for DSN1PRNT.

You can specify the PRINT parameter with or without page range specifications. If you do not specify a range, all pages of the SYSUT1 are printed. If you want to limit the range of pages that are printed, you can do so by indicating the beginning and ending page numbers with the PRINT parameter or, if you want to print a single page, by indicating only the beginning page. In either case, your range specifications must be from one to eight hexadecimal characters in length.

The following example shows how to code the PRINT parameter if you want to begin printing at page X'2F0' and to stop at page X'35C':
PRINT(2F0,35C)

The relationship between the page size and the number of pages in a 4-GB data set is shown in the following table.

Table 135. Relationship between page size and the number of pages in a 4-GB data set

<table>
<thead>
<tr>
<th>Page size</th>
<th>Number of pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 KB</td>
<td>X'100000'</td>
</tr>
<tr>
<td>8 KB</td>
<td>X'80000'</td>
</tr>
<tr>
<td>16 KB</td>
<td>X'40000'</td>
</tr>
<tr>
<td>32 KB</td>
<td>X'20000'</td>
</tr>
</tbody>
</table>

For example, if PAGESIZE is 4 KB, the page number of the first page of the third data set is 2*X'100000' = X'200000'.

To print only the header page for a nonpartitioned table space, specify PRINT(0).

You can indicate the format of the row data in the PRINT output by specifying EBCDIC, ASCII, or UNICODE. The part of the output that is affected by these options is in bold in the following figure.

Figure 111. The part of the DSN1PRNT FORMAT output that is affected by the EBCDIC, ASCII, and UNICODE options
**EBCDIC**
Indicates that the row data in the PRINT output is to be displayed in EBCDIC.

The default value is **EBCDIC** if the first page of the input data set is not a header page.

If the first page is a header page, DSN1PRNT uses the format information in the header page as the default format. However, if you specify EBCDIC, ASCII, or UNICODE, that format overrides the format information in the header page. The unformatted header page dump is always displayed in EBCDIC, because most of the fields are in EBCDIC.

**ASCII**
Indicates that the row data in the PRINT output is to be displayed in ASCII. Specify ASCII when printing table spaces that contain ASCII data.

**UNICODE**
Indicates that the row data in the PRINT output is to be displayed in Unicode. Specify UNICODE when printing table spaces that contain Unicode data.

**VALUE**
Causes each page of the input data set SYSUT1 to be scanned for the character string that you specify in parentheses following the VALUE parameter. Each page that contains that character string is then printed in SYSPRINT. You can specify the VALUE parameter in conjunction with any of the other DSN1PRNT parameters.

* (string)  
Can consist of from 1 to 20 alphanumeric EBCDIC characters. For non-EBCDIC characters, use hexadecimal characters.

* (hexadecimal-constant)  
Consists of from 2 to 40 hexadecimal characters. You must specify two apostrophe characters before and after the hexadecimal character string.

If, for example, you want to search your input file for the string '12345', your JCL should look like the following JCL:
```jcl
//STEP1 EXEC PGM=DSN1PRNT,PARM='VALUE(12345)'
```

Alternatively, you might want to search for the equivalent hexadecimal character string. If you are processing Unicode or ASCII input files, you must specify the string in hexadecimal. Your JCL should look like the following JCL:
```jcl
//STEP1 EXEC PGM=DSN1PRNT,PARM='VALUE(113132333435)'
```

**FORMAT**
Causes the printed output to be formatted. Page control fields are identified, and individual records are printed. Empty fields are not displayed.

**EXPAND**
Specifies that the data is compressed and causes DSN1PRNT to expand it before formatting. This option is intended to be used only under the direction of IBM Support.

When DSN1PRNT is run with the FORMAT EXPAND option, and the input data sets constitute a full image copy, the input data sets need to contain all pages of the original table space, including all dictionary pages. FORMAT EXPAND cannot be specified if the INCRCOPY or INLCOPY options are specified.
NODATA
Suppresses printing of table row data. The row headers are formatted and printed. Specify NODATA to reduce the volume of the output when the contents of the rows are not important.

NODATPGS
Suppresses all data pages of a table space. Specify NODATPGS to format and print only non-data pages to reduce the volume of the output when only certain page types are of interest (for example, LOB space map pages). Alternatively, you can specify NODHDR.

DSN1PRNT cannot format a leaf or nonleaf page for an index page set that contains keys with altered columns. When it encounters this situation, DSN1PRNT generates the following message:

*KEY WITH ALTERED COLUMN HAS BEEN DETECTED–UNABLE TO FORMAT PAGE*

DSN1PRNT generates unformatted output for the page.

FORMAT attempts to format a broken page and dumps the unformatted version of the page following the formatted version.

Related concepts:
“Using VERIFY with REPLACE and DELETE operations” on page 697

Printing with DSN1PRNT instead of DSN1COPY

If you want to print information about a data set, use the DSN1PRNT utility rather than the DSN1COPY utility. DSN1COPY scans the entire SYSUT1 data set, but DSN1PRNT might be able to stop scanning before the end of the data set. Also, the DSN1PRNT utility can write a formatted dump.

Determining the page size and data set size for DSN1PRNT

Before you run the DSN1PRNT utility, you must determine the page size and data set size (DSSIZE) for the page set.

Procedure

To determine the page size and data set size:

Issue a query against the Db2 catalog. For example, the query that is shown in the following figure returns this information for the DEPT table:

```
SELECT T.CREATOR, T.NAME, S.NAME AS TABLESPACE, S.PARTITIONS, S.PGSIZE,
     CASE S.DSSIZE
     WHEN 0 THEN
       CASE WHEN S.TYPE = 'G' THEN 4194304
            WHEN S.TYPE = 'L' THEN 4194304
            WHEN S.TYPE = 'O' THEN 4194304
            WHEN S.TYPE = 'P' THEN 4194304
            WHEN S.TYPE = 'R' THEN 4194304
       ELSE
         CASE WHEN S.PARTITIONS > 254 THEN
           CASE WHEN S.PGSIZE = 4 THEN 4194304
                WHEN S.PGSIZE = 8 THEN 8388608
                WHEN S.PGSIZE = 16 THEN 16777216
                WHEN S.PGSIZE = 32 THEN 33554432
           ELSE NULL
         END
       END
     WHEN S.PARTITIONS > 64 THEN 4194304
     WHEN S.PARTITIONS > 32 THEN 1048576
```
WHEN S.PARTITIONS > 16 THEN 2097152
WHEN S.PARTITIONS = 0 THEN 4194304
ELSE 2097152
END
ELSE S.DSSIZE
END
AS DSSIZE
FROM SYSIBM.SYSTABLES T,
SYSIBM.SYSTABLESPACE S
WHERE
T.NAME = 'DEPT' AND
T.TSNAME = S.NAME;

Related reference:
"Data sets that REORG INDEX uses" on page 535

Sample DSN1PRNT control statements

Use the sample control statements as models for developing your own DSN1PRNT control statements.

Example 1: Printing a data set and formatting the output

The following example specifies that the DSN1PRNT utility is to print the data set that is identified by the SYSUT1 DD statement and the output is to be formatted. This data set is to be printed on the data set that is identified by the SYSPRINT DD statement. The fifth-level qualifier in the data set name can be either I0001 or J0001. This example uses I0001.

Example 2: Printing a nonpartitioning index with a 64-MB piece size

The following example specifies that DSN1PRNT is to print the first 16 pages of the 61st piece of an nonpartitioned index with a piece size of 64 MB. The pages that are to be printed are identified by the PRINT option. These page values are determined as follows: A data set of size 64 MB contains X'4000' 4-KB pages. Decimal 61 is X'3D'. The page number of the first page of the 61st piece is 4000*(3D-1) = 4000*3C = F0000. To print the last 16 pages of the 61st piece, specify PARM=(PRINT(F3FF0,F3FFF), ...).

Example 3: Printing a single page of an image copy

The following example specifies that DSN1PRNT is to print one page of an image copy. The image copy is identified by the SYSUT1 DD statement. The PRINT option specifies that the only page to be printed is X'1'.
Example 4: Printing a partitioned data set

The following example specifies that DSN1PRNT is to print the data set that is identified by the SYSUT1 DD statement. Because this data set is a table space that was defined with the LARGE option, the DSSIZE(4G) option is specified in the parameter list for DSN1PRNT. You could specify the LARGE option in this list instead, but specifying DSSIZE(4G) is recommended. This input table space has 260 partitions, as indicated by the NUMPARTS option.

```
//RUNPRNT1 EXEC PGM=DSN1PRNT,
//       PARM='DSSIZE(4G),PRINT,NUMPARTS(260),FORMAT'
//STEPLIB DD DSN=DB2A.SDSNLOAD,DISP=SHR
//SYSUT1 DD DSN=DSNCAT.DSNDBC.DBOM0301.TPOM0301.I0001.A008,DISP=SHR
/*
```

Example 5: Printing a page range of a specific partition

It is difficult to determine page numbers for a specific partition of a partitioned table space.

This example describes a simple way of using DSN1PRNT without needing to calculate page numbers.

1. Run DSN1PRNT on the partition you want to process without specifying a PRINT range. Set the FORMAT option to NODATPGS. Data pages are not printed reducing the use of spool space.

   ```
   // EXEC PGM=DSN1PRNT,
   //       PARM='FORMAT,NODATPGS,NUMPARTS(8)'
   //SYSUT1 DD DSN=DSNT6USR.DSNDBC.V9DS306.XV9D0000.I0001.A008,DISP=SHR
   ```

   The printout includes page numbers. Use these page numbers to setup another DSN1PRNT job using the appropriate page numbers.

2. Run DSN1PRNT on partition 8 specifying your PRINT range

   ```
   // EXEC PGM=DSN1PRNT,
   //       PARM='PRINT(xx000000,xx000020),FORMAT,NUMPARTS(8)'
   //SYSUT1 DD DSN=DSNT6USR.DSNDBC.V9DS306.XV9D0000.I0001.A008,DISP=SHR
   ```

   The page range must be specified in hexadecimal format.

Example 6: Specifying Unicode output for DSN1PRNT

When you specify the UNICODE option for DSN1PRNT, you are not going to see non-Latin Unicode characters, such as Japanese characters, in your output. When you specify the UNICODE option, DSN1PRNT takes the hexadecimal data and formats it as ASCII instead of the default EBCDIC.

A problem might arise when the data that you want DSN1PRNT to handle is in UTF-16. In the case of UTF-16 data, DSN1PRNT takes only the second byte of the data and formats that part of the data as ASCII. Thus, the output might not be correct. For example, the UTF-16 hexadecimal values X'0030' and X'1130' are both output as 0, because the first byte of each ("00" and "11" respectively) is ignored.

1018 Utility Guide and Reference
The remaining part ("30") is interpreted as an ASCII 0. In UTF-16, X'0030' is the hexadecimal value for 0, but X'1130' is the hexadecimal value for a Hangul character.

In the following DSN1PRNT example, notice the three bold hexadecimal values: X'0041', X'0141', and X'0241'. The output for all three of these values is A.A.A, even though they each correspond to different characters in UTF-16. (X'0041' is A, X'0141' is ⍮, and X'0241' is the Latin capital character for glottal stop.)

Related information:

UTFs (Db2 Internationalization Guide)
Chapter 47. DSN1SDMP

IBM Support might advise you to use the IFS selective dump (DSN1SDMP) stand-alone utility. DSN1SDMP enables you to force dumps when selected Db2 trace events occur, write Db2 trace records to user-defined z/OS data sets, or start another Db2 trace.

To ensure that you do not take action on an IFCID 4 or IFCID 5 start or stop trace record, it is good practice to add

```plaintext
P4,00
DR,04,X'hhhh'
```

to your control statement, where hhhh is the hex representation of the IFCID that you are trying to trigger on.

**Output**

One intended use of this utility is to aid in determining and correcting system problems. When diagnosing Db2, you might need to refer to licensed documentation to interpret output from this utility.

**Environment**

Run DSN1SDMP as a z/OS job, and execute it with the DSN TSO command processor. To execute DSN1SDMP, the Db2 subsystem must be running.

The z/OS job completes only under one of the following conditions:

- The TRACE and any additional selection criteria that are started by DSN1SDMP meet the criteria specified in the FOR parameter.
- The TRACE that is started by DSN1SDMP is stopped by using the STOP TRACE command.
- The job is canceled by the operator.

If you must stop DSN1SDMP, use the STOP TRACE command.

**Authorization required**

To execute this utility, the privilege set of the process must include one of the following privileges or authorities:

- TRACE system privilege
- SYSOPR authority
- SYSADM authority
- MONITOR1 or MONITOR2 privileges (if you are using user-defined data sets)
- SQLADM authority
- System DBADM authority
- SECADM authority

The user who executes DSN1SDMP must have EXECUTE authority on the plan that is specified in the `trace-parameters` of the START TRACE keyword.
Required data sets

DSN1SDMP uses the following DD statements:

**SDMPIN**
Defines the control data set that specifies the input parameters to DSN1SDMP. This DD statement is required. The LRECL is 80. Only the first 72 columns are checked by DSN1SDMP.

**SDMPPRNT**
Defines the sequential message data set that is used for DSN1SDMP messages. If the SDMPPRNT DD statement is omitted, no messages are written. The LRECL is 131.

**SYSABEND**
Defines the data set that is to contain an ABEND dump in case DSN1SDMP abends. This DD statement is optional.

**SDMPTRAC**
Defines the sequential Db2 trace record data set that Db2 returns to DSN1SDMP. The DD statement is required only if trace data is written to an OPX trace destination. If the destination is anything other than an OPX buffer, SDMPTRAC is ignored.

Trace records that Db2 writes to SDMPTRAC are of the same format as SMF or GTF records except that the SDMPTRAC trace record headers contain the monitor header (that is mapped by DSNDQWIW). The DCB parameters are VB, BLKSIZE=32760, LRECL=32756.

**SYSTSIN**
Defines the DSN commands to connect to Db2 and to execute an IFC selective dump:

```
DSN SYSTEM(subsystem name)
RUN PROG(DSN1SDMP) LIB('prefix.SDSNLOAD') PLAN(DSNEDCL)
```

The Db2 subsystem name must be filled in by the user. The DSN RUN command must specify a plan for which the user has execute authority. DSN1SDMP dump does not execute the specified plan; the plan is used only to connect to Db2.

When no plan name is specified on the DSN RUN command, the default plan name is the program name. When DSN1SDMP is executed without a plan, DSN generates an error if no DSN1SDMP plan exists for which the user has execute authority.

Related concepts:

- [Obtaining Db2 Diagnosis Guide and Reference](#)

Related reference:

- [Trace data record format (Db2 Performance)](#)

Syntax and options of the DSN1SDMP control statement

The DSN1SDMP utility control statement, with its multiple options, defines the function that the utility job performs.
### Option descriptions

**START TRACE** *(trace-parameters)*

Indicates the start of a DSN1SDMP job. START TRACE is a required keyword and must be the first keyword that is specified in the SDMPIN input stream.

If the START TRACE command in the SDMPIN input stream is not valid, or if the user is not properly authorized, the IFI (instrumentation facility interface) returns an error code and START TRACE does not take effect. DSN1SDMP writes the error message to the SDMPPRNT data set.

Notes:
1. The options in the *second-trace-spec* do not have to be specified immediately following the STTRACE option. However, they can be specified only if the STTRACE option is also specified.
**Trace Destination:** If Db2 trace data is to be written to the SDMPTRAC data set, the trace destination must be an IFI online performance (OP) buffer. OP buffer destinations are specified in the DEST keyword of START TRACE. Eight OP buffer destinations exist, OP1 to OP8. The OPX trace destination assigns the next available OP buffer. Any record destined for the exclusive internal trace table (RES) is not eligible to be evaluated. For example, if you start IFCID(0) DEST(RES), this will not execute DSN1SDMP logic and cannot be acted upon.

The Db2 output text from the START TRACE command is written to SDMPPRNT.

START TRACE and its associated keywords must be specified first. Specify the remaining selective dump keywords in any order following the START TRACE command.

**SELECT function,offset,data-specification**

Specifies selection criteria in addition to those that are specified on the START TRACE command. SELECT expands the data that is available for selection in a trace record and allows more specific selection of data in the trace record than using START TRACE alone. You can specify a maximum of eight SELECT criteria.

The selection criteria use the concept of the current-record pointer. Db2 initializes the current-record pointer to zero, that is, at the beginning of the trace record. For this instance of the DSN1SDMP trace, the trace record begins with the self-defining section. The current-record pointer can be modified by Px and LN functions, which are described in the list of functions below.

You can specify the selection criteria with the following parameters:

**function**

Specifies the type of search that is to be performed on the trace record. The specified value must be two characters. The possible values are:

- **DR** Specifies a direct comparison of data from the specified offset. The offset is always calculated from the current-record pointer.

- **GE** Specifies a comparison of data that is greater than or equal to the value of the specified offset. The offset is always calculated from the current-record pointer. The test succeeds if the data from the specified offset is greater than or equal to *data-specification*, which you can specify on the SELECT option.

- **LE** Specifies a comparison of data that is less than or equal to the value of the specified offset. The offset is always calculated from the current-record pointer. The test succeeds if the data from the specified offset is less than or equal to *data-specification*, which you specify on the SELECT option.

- **P1, P2, or P4**

  Selects the 1-, 2-, or 4-byte field that is located *offset* bytes past the start of the record. The function then moves the current-record pointer that number of bytes into the record. P1, P2, and P4 always start from the beginning of the record (plus the offset that you specify).

  This offset is saved as the current-record pointer that is to be used on subsequent DR, LE, GR, and LN requests.

  For example, suppose that the user knows that the offset to the standard header is 4 bytes long and is located in the first 4 bytes of the record. P4,00 reads that offset and moves the current-record pointer to the start of the standard header.
LN  Advances the current-record pointer by the number of bytes that are indicated in the 2-byte field that is located offset bytes from the previous current-record pointer.

This offset is saved as the current-record pointer that is to be used on subsequent DR, LE, GR, and LN requests.

offset
Specifies the number (in decimal) of bytes into the trace record where the comparison with the data-specification field begins. The offset starts from the beginning of the trace record after a P1, P2, or P4, and from the current-record pointer after a GE, LE, LN, or DR.

The format of the Db2 trace record at data-specification comparison time is shown in the following figure.

Figure 112. Format of the Db2 trace record at data specification comparison time

- The format of the self-defining section depends on the trace type.
- The format and content of the data sections depend on the IFCID that is being recorded. Each record can have one or more data sections. Each data section can have multiple repeating groups.
- The format and content of the trace header section depends on the trace type.

data-specification
Specifies that the data can be hexadecimal (for example, X'9FECBA10') or character (C'FIELD').

ACTION
Specifies the action to perform when a trace record passes the selection criteria of the START TRACE and SELECT keywords.

Attention: The purpose of the ACTION keyword is to facilitate problem analysis. You should use it with extreme caution because you might damage existing data. Not all abends are recoverable, even if the ABENDRET parameter is specified. Some abends might force the Db2 subsystem to terminate, particularly those abends that occur during end-of-task or end-of-memory processing due to the agent having experienced a previous abend.

action(abend-code)
Specifies a particular action to perform. Possible values for action are:

ABENDRET
ABEND and retry the agent.

ABENDTER
ABEND and terminate the agent.

An abend reason code can also be specified on this parameter. The codes must be in the range X'00E60100' to X'00E60199'. The default value is X'00E60100'.

STTRACE
Specifies that a second trace is to be started when a trace record passes the selection criteria.
If you do not specify action or STTRACE, the record is written and no action is performed.

**AFTER**(integer)

Specifies that the ACTION is to be performed after the trace point is reached integer times.

integer must be between 1 and 32767.

The default value is **AFTER(1)**.

**FOR**(integer)

Specifies the number of times that the ACTION is to take place when the specified trace point is reached. After integer times, the trace is stopped, and DSN1SDMP terminates.

integer must be between 1 and 32767 and includes the first action. If no SELECT criteria are specified, use an integer greater than 1; the START TRACE command automatically causes the action to take place one time.

The default value is **FOR(1)**.

**ACTION2**

Specifies the action to perform when a trace record passes the selection criteria of the START TRACE, SELECT, and SELECT2 keywords.

Attention: The ACTION2 keyword, like the ACTION keyword, should be used with extreme caution, because you might damage existing data. Not all abends are recoverable, even if the ABENDRET parameter is specified. Some abends might force the Db2 subsystem to terminate, particularly those that occur during end-of-task or end-of-memory processing due to the agent having experienced a previous abend.

**action**(abend-code)

Specifies a particular action to perform. Possible values for action are:

**ABENDRET**

ABEND and retry the agent.

**ABENDTER**

ABEND and terminate the agent.

An abend reason code can also be specified on this parameter. The codes must be in the range X'00E60100-00E60199'. If no abend code is specified, X'00E60100' is used.

If you do not specify action, the record is written and no action is performed.

**FILTER**

Specifies that DSN1SDMP is to filter the output of the second trace based on either an ACE or an EB.

**(ACE)**

Specifies that DSN1SDMP is to include trace records only for the agent control element (ACE) that is associated with the agent when the first action is triggered and the second trace is started.

**(EB)**

Specifies that DSN1SDMP is to include trace records only for the execution block (EB) that is associated with the agent when the first action is triggered and the second trace is started.
**COMMAND**
Indicates that the specified command is to be issued when a trace record passes the selection criteria for the first trace and a second trace is started. You can start a second trace by specifying the STTRACE option.

`command`
Specifies a specific command to be issued.

**FOR2(integer)**
Specifies the number of times that the ACTION2 is to take place when the specified second trace point is reached. After `integer` times, the second trace is stopped, and DSN1SDMP terminates.

`integer` must be between 1 and 32767 and includes the first action. If no SELECT2 criteria are specified, use an integer greater than 1; the STTRACE option automatically causes the action to take place one time.

The default value is `FOR2(1)`.

**AFTER2(integer)**
Specifies that the ACTION2 is to be performed after the second trace point is reached `integer` times.

`integer` must be between 1 and 32767.

The default value is `AFTER2(1)`.

**SELECT2 function,offset,data-specification**
Specifies selection criteria for the second trace. This option functions like the SELECT option, except that it pertains to the second trace only. You can start a second trace by specifying the STTRACE option.

**Related reference:**
- START TRACE (Db2) (Db2 Commands)
- Trace field descriptions (Db2 Performance)

---

**Assigning buffers**
You must specify the OPX destination for all traces that are being recorded to an OPn buffer for the DSN1SDMP utility to use. By specifying the OPX destination, you avoid the possibility of starting a trace to a buffer that is already assigned.

If a trace is started to an OPn buffer that has already been assigned, DSN1SDMP waits indefinitely until the trace is manually stopped. The default for MONITOR-type traces is the OPX destination (the next available OP buffer). Other trace types must be explicitly directed to OP destinations via the DEST keyword of the START TRACE command. DSN1SDMP interrogates the IFCAOPN field after the START TRACE COMMAND call to determine if the trace was started to an OP buffer.

**Trace Destination:** If Db2 trace data is to be written to the SDMPTRAC data set, the trace destination must be an IFI online performance (OP) buffer. OP buffer destinations are specified in the DEST keyword of START TRACE. Eight OP buffer destinations exist, OP1 to OP8. The OPX trace destination assigns the next available OP buffer. Any record destined for the exclusive internal trace table (RES) is not eligible to be evaluated. For example, if you start IFCID(0) DEST(RES), this will not execute DSN1SDMP logic and cannot be acted upon.
Trace records are written to the SDMPTRAC data set when the trace destination is an OP buffer. The instrumentation facilities component (IFC) writes trace records to the buffer and posts DSN1SDMP to read the buffer when it fills to half of the buffer size.

You can specify the buffer size on the BUFSIZE keyword of the START TRACE command. All returned records are written to SDMPTRAC.

If the number of generated trace records requires a larger buffer size than was specified, you can lose some trace records. If this happens, error message DSN2724I is issued.

Conditions for generating a dump

DSN1SDMP generates a Db2 dump when certain events occur.

DSN1SDMP generates a Db2 dump when all of the following events occur:

- Db2 produces a trace record that satisfies all of the selection criteria.
- You specify an abend action (ABENDRET or ABENDTER).
- The AFTER and FOR conditions for the trace are satisfied.

If all three events occur, an 00E601xx abend occurs. xx is an integer between 1 and 99 that Db2 obtains from the user-specified value on the ACTION keyword.

Stopping or modifying DSN1SDMP traces

You can stop and modify DSN1SDMP traces.

Procedure

To stop the DSN1SDMP utility:

Issue a STOP TRACE command. For example, if the DSN1SDMP utility does not finish, you can might stop it by issuing the following command:

```
-STOP TRACE=P CLASS(32)
```

DSN1SDMP executes as a stand-alone batch utility without requiring external intervention from the console operator or other programs. During execution, DSN1SDMP issues an IFI READA request to obtain the data from the OPn buffer and a STOP TRACE command to terminate the original trace that is started by DSN1SDMP.

A STOP TRACE or MODIFY TRACE command that is entered from a console for the trace that is started by DSN1SDMP causes immediate abnormal termination of DSN1SDMP processing. The IFI READA function terminates with an appropriate IFI termination message and reason code. Additional error messages and reason codes that are associated with the DSN1SDMP STOP TRACE command vary depending on the specific trace command that is entered by the console operator. If the console operator terminates the original trace by using the STOP TRACE command, the subsequent STOP TRACE command that is issued by DSN1SDMP fails.

If the console operator enters a MODIFY TRACE command and processing of this command completes before the STOP TRACE command is issued by DSN1SDMP, the modified trace is also terminated.

Related reference:

- [STOP TRACE (Db2) (Db2 Commands)]
Sample DSN1SDMP control statements

Use the sample control statements as models for developing your own DSN1SDMP control statements.

Example 1: Creating the JCL for DSN1SDMP

This example shows the skeleton JCL for a DSN1SDMP job.

```bash
//DSN1J018 JOB 'IFC SD',CLASS=A,
// MSGLEVEL=(1,1),USER=SYSADM,PASSWORD=SYSADM,REGION=1024K
//******************************************************************************
//*                        THIS IS A SKELETON OF THE JCL USED TO RUN DSN1SDMP.           *
//*                        YOU MUST INSERT SDMPIN DD.                                *
//*******************************************************************************
//IFCSD EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//STEPLIB DD DISP=SHR,DSN=prefix.SDSNLOAD
//SYSPRINT DD SYSOUT**
//SYSTSPRT DD SYSOUT**
//SDMPTRAC DD DISP=(NEW,CATLG,CATLG),DSN=IFCSD.TRACE,
// UNIT=SYSDA,SPACE=(B192,(100,100)),DB=(DSORG=PS,
// LRECL=32756,RECFM=VB,BLKSIZE=32760)
//SDMPIN DD *
//******************************************************************************
//*                        INSERT SDMPIN DD HERE. IT MUST BEGIN WITH A VALID          *
//*                        START TRACE COMMAND (WITHOUT THE SUBSYSTEM RECOGNITION CHAR)
//*******************************************************************************
//(VALID SDMPIN GOES HERE)
/*
//******************************************************************************
//SYSUDUMP DD SYSOUT**
//SYSTIN DD *
DSN SYSTEM(DSN)
RUN PROG(DSN1SDMP) PLAN(DSNEDCL)
END
//*
```

Example 2: Abending and retrying agent on -904 SQL CODE

This example specifies that Db2 is to start a performance trace (which is indicated by the letter A) and activate IFCID 53, 58. To start only those IFCIDs that are specified in the IFCID option, use trace classes 30-32. In this example, trace class 32 is specified. The IFCID 53 and 58 are started and inspected to see if they match the SELECT criteria.

The SELECT option indicates additional criteria for data in the trace record. In this example, the P4,00 positions the current record pointer to the product section. The GE, 04,'X'0005' ensures that the IFCID being traced is either an IFCID 53 or 58 and is not an IFCID4 which is automatically generated via the START TRACE command. The P4,08 positions the current record pointer to data section 1 of the IFCID 53 or 58. A direct comparison is then made at decimal offset 74 for SQL code X'FFFFFFFF78'.
When a trace record passes the selection criteria of the START TRACE command and SELECT keywords, DSN1SDMP is to perform the action that is specified by the ACTION keyword. In this example, the job is to abend and try again with reason code 00E60188. This action is to take place only once, as indicated by the FOR option. FOR(1) is the default, and is therefore not required to be explicitly specified.

```
//SDMPIN DD *
START TRACE=A CLASS(32) IFCID(53,58) DEST(OPX)
   FOR(1)
   AFTER(1)
   ACTION(ABENDRET(00E60188))
SELECT
 * Position to the product section
  P4,00
 * Ensure QWHSIID = 58 or 53 (not IFCID 4)
  GE,04,X'0005'
 * Position to the data section 1
  P4,08
 * Compare SQLCODE in QW0058SQ or QW0053SQ
  DR,74,X'FFFFFC78'
/*
```

Figure 113. Example job that abends and terminates agent on -904 SQL code

**Example 3: Abending and retrying on RMID 20**

This example specifies that Db2 is to start a performance trace (which is indicated by the letter P) and activate all IFCIDs in classes 3 and 8. The trace output is to be recorded in a generic destination that uses the first free OPn slot, as indicated by the DEST option. The TDATA (TRA) option specifies that a CPU header is to be placed into the product section of each trace record.

The SELECT option indicates additional criteria for data in the trace record. In this example, the SELECT option first specifies that the current-record pointer is to be placed at the 4-byte field that is located at the start of the record. The current record pointer is then to be advanced the number of bytes that are indicated in the 2-byte field that is located at the current record pointer. The utility is then to directly compare the data that is 4 bytes from the current-record pointer with the value X'0025'.

When a trace record passes the selection criteria of the START TRACE command and SELECT keywords, DSN1SDMP is to perform the action that is specified by the ACTION keyword. In this example, the job is to abend and retry the agent.
Example 4: Generating a dump on SQLCODE -811 RMID16 IFCID 5

This example specifies that Db2 is to start a performance trace (which is indicated by the letter P) and activate all IFCIDs in class 3. The trace output is to be recorded in the system management facility (SMF). The TDATA (COR,TRA) option specifies that a trace header and a CPU header are to be placed into the product section of each trace record.

The SELECT option indicates additional criteria for data in the trace record. In this example, the SELECT option first specifies that the current-record pointer is to be placed at the 4-byte field that is located at the start of the record. The utility is then to directly compare the data that is 2 bytes from the current-record pointer with the value X'0116003A'. The current record pointer is then to be moved to the 4-byte field that is located 8 bytes past the start of the current record. The utility is then to directly compare the data that is 74 bytes from the current-record pointer with the value X'FFFFFFCD5'.

When a trace record passes the selection criteria of the START TRACE command and SELECT keywords, DSN1SDMP is to perform the action that is specified by the ACTION keyword. In this example, the job is to abend with reason code 00E60188 and retry the agent. This action is to take place only once, as indicated by the FOR option. FOR(1) is the default, and is therefore not required to be explicitly specified. AFTER(1) indicates that this action is to be performed the first time the trace point is reached. AFTER(1) is also the default.
Example 5: Starting a second trace

This example job starts a trace on IFC 196 records. An IFC 196 record is written when a lock timeout occurs. In this example, when a lock timeout occurs, DSN1SDMP is to start a second trace, as indicated by the ACTION(STTRACE) option. This second trace is to be an accounting trace, as indicated by the COMMAND START TRACE(ACCTG) option. This trace is to include records only for the ACE that is associated with the agent that timed out, as indicated by the FILTER(ACE) option. When the qualifying accounting record is found, DSN1SDMP generates a dump.

```sql
//SDMPIN DD *
START TRACE=P CLASS(3) RMID(22) DEST(SMF) TDATA(COR,TRA)
AFTER(1)
FOR(1)
SELECT
* POSITION TO HEADERS (QWHS IS ALWAYS FIRST)
  P4,00
* CHECK QWHS 01, FOR RMID 16, IFCID 58
  DR,02,X'0116003A'
* POSITION TO SECOND SECTION (1ST DATA SECTION)
  P4,08
* COMPARE SQLCODE FOR 811
  DR,74,X'FFFFFFD5'
  ACTION(ABENDRET(00E60188))
/*
Figure 115. Example job that generates a dump on SQL code -811 RMID16 IFCID

Example 5: Starting a second trace

This example job starts a trace on IFC 196 records. An IFC 196 record is written when a lock timeout occurs. In this example, when a lock timeout occurs, DSN1SDMP is to start a second trace, as indicated by the ACTION(STTRACE) option. This second trace is to be an accounting trace, as indicated by the COMMAND START TRACE(ACCTG) option. This trace is to include records only for the ACE that is associated with the agent that timed out, as indicated by the FILTER(ACE) option. When the qualifying accounting record is found, DSN1SDMP generates a dump.

```sql
//SDMPIN DD *
* START ONLY IFCID 196, TIMEOUT
START TRACE=P CLASS(32) IFCID(196) DEST(SMF)
AFTER(1)
* ACTION = START ACCOUNTING TRACE
 ACTION(STTRACE)
* FILTER ON JUST 196 RECORDS...
 SELECT
 P4,00
 DR,04,X'00C4'
* WHEN ACCOUNTING IS CUT, ABEND
 ACTION2(ABENDRET(00E60188))
* START THE ACCOUNTING TRACE FILTER ON THE ACE OF THE AGENT
* THAT TIMED OUT
 COMMAND
 START TRACE(ACCTG) CLASS(32) IFCID(3) DEST(SMF)
* Filter can be for ACE or EB
 FILTER(ACE)
/*
Figure 116. Example job that starts a second trace.

Related reference:

[STOP TRACE (Db2) (Db2 Commands)](navigable_url)
Part 4. Appendixes
Appendix A. Supplied stored procedures for utility operations

Db2 provides some stored procedures that you can call in application programs to perform a number of utility functions. Typically, these procedures are created during installation or migration.

The following supplied stored procedures are particularly useful for utility operations:

- DSNUTILU stored procedure (Db2 SQL)
- DSNACCOX stored procedure (Db2 SQL)

For the complete list of stored procedures that are provided with Db2, see Procedures that are supplied with Db2 (Db2 SQL).

Related concepts:

- Sample callers of utilities stored procedures (Db2 Installation and Migration)
- Migration step 22: Configure Db2 for running stored procedures and user-defined functions (optional) (Db2 Installation and Migration)

Related tasks:

- Chapter 6, “Invoking Db2 online utilities,” on page 15
- Migration step 23: Set up Db2-supplied routines (Db2 Installation and Migration)
- Installing Db2-supplied routines during installation (Db2 Installation and Migration)
- Implementing Db2 stored procedures ()
Appendix B. Advisory or restrictive states

To control access and help ensure data integrity, Db2 sets a restrictive or nonrestrictive (advisory) status on certain objects. However, you can take steps to correct each status.

Use the **DISPLAY DATABASE** command to display the current status for an object.

In addition to the states mentioned in this topic, the output from the **DISPLAY DATABASE** command might also indicate that an object is in logical page list (LPL) status. This state means that the pages that are listed in the LPL PAGES column are logically in error and are unavailable for access. Db2 writes entries for these pages in an LPL.

**Related tasks:**
- Removing pages from the logical page list (Db2 Administration Guide)

**Related reference:**
- **DISPLAY DATABASE (Db2) (Db2 Commands)**

Auxiliary CHECK-pending status

When the auxiliary CHECK-pending status is set on a base table space, that base table space is unavailable for processing by SQL.

The auxiliary CHECK-pending (ACHKP) restrictive status is set on when at least one base table LOB column error is detected and not invalidated as a result of running **CHECK DATA AUXERROR REPORT**. A base table space with the **NOT LOGGED** attribute and its LOB table spaces, which also have the **NOT LOGGED** attribute, are recovered to the current point in time in the same **RECOVER** utility invocation. The base table space is put in the auxiliary CHECK-pending state.

An XML table space is set to auxiliary CHECK-pending when **CHECK DATA** is run with the **XMLERROR REPORT** option, and **CHECK DATA** finds an error in an XML table space, the corresponding base table space, or an index space for the node ID.

Refer to the following table for information about resetting the auxiliary CHECK-pending status. This table lists the status name, abbreviation, affected object, and any corrective actions.

<table>
<thead>
<tr>
<th>Status</th>
<th>Abbreviation</th>
<th>Object affected</th>
<th>Corrective action</th>
<th>Notes</th>
</tr>
</thead>
</table>
| Auxiliary CHECK-pending    | ACHKP        | Base table space  | 1. Update or delete invalid LOBs and XML objects using SQL.  
2. Run the **CHECK DATA** utility with the appropriate **SCOPE** option to verify the validity of LOBs and XML objects and reset ACHKP status.  
You can use the **REPAIR** utility, followed by **CHECK DATA**, to reset the ACHKP status, but use caution. | 1     |

© Copyright IBM Corp. 1983, 2017
Table 136. Resetting auxiliary CHECK-pending status (continued)

<table>
<thead>
<tr>
<th>Status</th>
<th>Abbreviation</th>
<th>Object affected</th>
<th>Corrective action</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td></td>
<td></td>
<td>1. A base table space in the ACHKP status is unavailable for processing by SQL.</td>
<td></td>
</tr>
</tbody>
</table>

**Auxiliary warning status**

A base table space or LOB table space in the auxiliary warning status remains available for processing by SQL.

The auxiliary warning (AUXW) status is set when at least one base table LOB column has an invalidated LOB as a result of running CHECK DATA AUXERROR INVALIDATE. An attempt to retrieve an invalidated LOB results in a -904 SQL return code.

A base table space with the NOT LOGGED attribute and its LOB table spaces, which also have the NOT LOGGED attribute, are recovered to the current point in time in the same RECOVER utility invocation. The LOB table spaces are put in the auxiliary warning state if there were updates to the LOB table spaces after the recoverable point.

The RECOVER utility also sets AUXW status if it finds an invalid LOB column. Invalid LOB columns might result from a situation in which all the following actions occur:
1. LOB table space was defined with LOG NO.
2. LOB table space was recovered.
3. LOB was updated since the last image copy.

Refer to the following table for information about resetting the auxiliary warning status. This table lists the status name, abbreviation, affected objects, and any corrective actions.

Table 137. Resetting auxiliary warning status

<table>
<thead>
<tr>
<th>Status</th>
<th>Abbreviation</th>
<th>Object affected</th>
<th>Corrective action</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auxiliary warning</td>
<td>AUXW</td>
<td>Base table space</td>
<td>1. Update or delete invalid LOBs and XML objects using SQL.</td>
<td>1,2,3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. If an orphan LOB exists or a version mismatch exists between the base table and</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the auxiliary index, use REPAIR to delete the LOB from the LOB table space.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. Run CHECK DATA utility to verify the validity of LOBs and XML objects and reset</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AUXW status.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Auxiliary warning</td>
<td>AUXW</td>
<td>LOB table space</td>
<td>1. Update or delete invalid LOBs and XML objects using SQL.</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. If an orphan LOB exists or a version mismatch exists between the base table and</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the auxiliary index, use REPAIR to delete the LOB from the LOB table space.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. Run CHECK LOB utility to verify the validity of LOBs and XML objects and reset</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AUXW status.</td>
<td></td>
</tr>
</tbody>
</table>
Table 137. Resetting auxiliary warning status (continued)

<table>
<thead>
<tr>
<th>Status</th>
<th>Abbreviation</th>
<th>Object affected</th>
<th>Corrective action</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A base table space or LOB table space in the AUXW status is available for processing by SQL, even though it contains invalid LOBs. However, an attempt to retrieve an invalid LOB results in a -904 SQL return code.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Db2 can access all rows of a base table space that are in the AUXW status. SQL can update the invalid LOB column and delete base table rows, but the value of the LOB column cannot be retrieved. If Db2 attempts to access an invalid LOB column, a -904 SQL code is returned. The AUXW status remains on the base table space even when SQL deletes or updates the last invalid LOB column.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>If CHECK DATA AUXERROR REPORT encounters only invalid LOB columns and no other LOB column errors, the base table space is set to the auxiliary warning status.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CHECK-pending status

The CHECK-pending (CHKP) restrictive status indicates that an object might be in an inconsistent state and must be checked.

The following utilities set CHECK-pending status on a table space if referential integrity constraints are encountered:
- LOAD with ENFORCE NO
- RECOVER to a point in time
- CHECK LOB

Table check violations place a table space or partition in CHECK-pending status when any of the following conditions exist:
- A check constraint is defined on a populated table by using the ALTER TABLE statement, and the value of the CURRENT RULES special register is Db2.
- The LOAD utility is run with CONSTRAINTS NO, and check constraints are defined on the table.
- CHECK DATA is run on a table that contains violations of check constraints.
- A point-in-time RECOVER introduces violations of check constraints.

An index might be placed in CHECK-pending status if you recovered an index to a specific RBA or LRSN from a copy and applied the log records, but you did not recover the table space in the same list. The CHECK-pending status can also be set on an index if you specified the table space and the index, but the recovery point in time was not a point of consistency (QUIESCE or COPY SHRLEVEL REFERENCE).

The CHECK-pending status can also affect a base table space or a LOB table space. CHECK-pending status for an XML table space is set only if a new XML schema for an XML type modifier was added or removed.

Db2 ignores informational referential integrity constraints and does not set CHECK-pending status for them.

The following table lists the actions to take to reset the CHECK-pending status.
### Table 138. Resetting CHECK-pending status

<table>
<thead>
<tr>
<th>Status</th>
<th>Abbreviation</th>
<th>Object affected</th>
<th>Corrective action</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHECK-pending</td>
<td>CHKP</td>
<td>Table space, base table space</td>
<td>Run the CHECK DATA utility. If a table space is in both REORG-pending and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CHECK-pending status (or auxiliary CHECK-pending status), run the REORG TABLESPACE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>utility first and then run CHECK DATA to reset the respective states.</td>
</tr>
<tr>
<td>CHECK-pending</td>
<td>CHKP</td>
<td>Partitioning index, nonpartitioning index, index, XML</td>
<td>1. Run the CHECK INDEX utility on the index.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>index on the auxiliary table</td>
<td>2. If any errors are found, use the REBUILD INDEX utility to rebuild the index</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>from existing data.</td>
</tr>
<tr>
<td>CHECK-pending</td>
<td>CHKP</td>
<td>LOB table space</td>
<td>Run the CHECK LOB utility to check the LOB table space. If any errors are found,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>take the following actions:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1. Correct any defects that are found in the LOB table space by using the REPAIR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>utility.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. Run CHECK LOB again to reset the CHECK-pending status.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. If the table space is in auxiliary warning (AUXW) status, take the actions in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>“Auxiliary warning status” on page 1038</td>
</tr>
<tr>
<td>CHECK-pending</td>
<td>CHKP</td>
<td>XML table space</td>
<td>Run CHECK DATA to check the XML table space. If any errors are found, take the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>following actions:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1. Correct any defects that are found in the XML table space by using the REPAIR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>utility.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. Run CHECK DATA again to reset the CHECK-pending status.</td>
</tr>
</tbody>
</table>

**Related reference:**

- “Auxiliary CHECK-pending status” on page 1037
- Chapter 11, “CHECK DATA,” on page 73
- Chapter 12, “CHECK INDEX,” on page 103
- Chapter 13, “CHECK LOB,” on page 121
- Chapter 28, “REORG TABLESPACE,” on page 553
- Chapter 29, “REBUILD INDEX,” on page 427
- Chapter 29, “REPAIR,” on page 669

## COPY-pending status

The COPY-pending (COPY) restrictive status indicates that the affected object must be copied.

Db2 ignores informational referential integrity constraints and does not set CHECK-pending status for them.

Refer to the following table for information about resetting the COPY-pending status. This table lists the status name, abbreviation, affected objects, and any corrective actions.
### Table 139. Resetting COPY-pending status

<table>
<thead>
<tr>
<th>Status</th>
<th>Abbreviation</th>
<th>Object affected</th>
<th>Corrective action</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>COPY-pending</td>
<td>COPY</td>
<td>Table space, table space partition</td>
<td>Take an image copy of the affected object.</td>
<td></td>
</tr>
</tbody>
</table>

### DBETE status

An object is a table space, table space partition, index space, index partition, or logical index partition. The DBETE status identifies the objects that need special attention by the user.

To reduce outages caused by certain DBET abends or page set access error abends during restart or RESTORE SYSTEM, Db2 tolerates such abends and puts objects into advisory status DBETE, as well as a restrictive exception status that requires objects to be recovered. Depending on the type of object, either a table space or an index space, the restrictive exception status can be RECP, RBDP, or PSRBD.

Refer to the following table for information about resetting the error status. This table lists the status name, abbreviation, affected objects, and any corrective actions.

### Table 140. Resetting OBJECT error status

<table>
<thead>
<tr>
<th>Status</th>
<th>Abbreviation</th>
<th>Object affected</th>
<th>Corrective action</th>
<th>Notes</th>
</tr>
</thead>
</table>
| OBJECT error   | DBETE        | Table space, table space partition, index space, index partition, or logical index partition | Issue -DISPLAY command to see the status of the object. If the object is in DBETE along with restrictive states such as RECP, RBDP, PSRBD, use one of the following utilities to recover or rebuild the object:  
• RECOVER utility  
• LOAD utility with the REPLACE option  
• REBUILD utility  
The DBETE status is reset when RECP, RBDP, or PSRBD status is reset by the utility.  
If a table space or index space that contains partitions has a status of DBETE and RECP and is also listed as being of type UN (unknown type), you can still use the utilities in the preceding list to recover or rebuild the entire space.  
DBETE, RECP, RBDP, and PSRBD status can also be reset by issuing the following command,  
-START DB(db name) SF(space name) ACCESS(FORCE).  
Contact IBM Support to report the problem.  
Db2 log records need to be analyzed to diagnose the cause of the problem and determine further actions. |                                            |
Group buffer pool RECOVER-pending status

The group buffer pool RECOVER-pending (GRECP) status is set on when a coupling facility fails with pages that were not externalized. The affected object must be recovered.

**Automatic recovery**

Db2 automatically initiates recovery of an index that is in GRECP or LPL status. Indexes become available immediately available after GRECP or LPL recovery.

In some rare cases when you run the RECOVER utility or RESTORE SYSTEM utility, an index might be left in REBUILD-pending status. In these cases, you must rebuild the index by running the REBUILD INDEX utility.

**Manual recovery**

Refer to the following table for information about resetting the group buffer pool RECOVER-pending status. This table lists the status name, abbreviation, affected objects, and any corrective actions.

<table>
<thead>
<tr>
<th>Status</th>
<th>Abbreviation</th>
<th>Object affected</th>
<th>Corrective action</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group buffer pool RECOVER-pending</td>
<td>GRECP</td>
<td>Object</td>
<td>Recover the object, or use START DATABASE to recover the object.</td>
<td></td>
</tr>
</tbody>
</table>

**Tip:** To avoid issuing -START DATABASE for every member, wait until a DSNI049I message is issued from each restarting member, then issue -START DATABASE to recover objects with GRECP-pending status. Any objects that are not defined with group buffer pool AUTOREC(YES) recovery need to be explicitly recovered.

**Informational COPY-pending status**

The informational COPY-pending (ICOPY) advisory status indicates that the affected object should be copied.

Refer to the following table for information about resetting the informational COPY-pending status. This table lists the status name, abbreviation, affected objects, and any corrective actions.

<table>
<thead>
<tr>
<th>Status</th>
<th>Abbreviation</th>
<th>Object affected</th>
<th>Corrective action</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Informational ICOPY</td>
<td>ICOPY</td>
<td>NOT LOGGED table spaces</td>
<td>Copy the affected table space.</td>
<td></td>
</tr>
<tr>
<td>Informational ICOPY</td>
<td>ICOPY</td>
<td>Partitioning index, nonpartitioning index, index on the auxiliary table</td>
<td>Copy the affected index.</td>
<td></td>
</tr>
</tbody>
</table>
### PRO restricted status

The Persistent Read Only (PRO) restricted status indicates that read access by SQL or utilities is allowed, but all updates to the table space partition are prohibited.

SQL updates or utilities that attempt to update a table space partition that is in PRO restricted status receive a resource unavailable error. When one or more table space partitions are in PRO restricted status, run utilities that update data at the partition level.

PRO restricted status is turned on to prohibit all updates by the system administrator or database administrator, or by a product that runs with Db2 for z/OS. The REPAIR SET utility sets the PRO restricted status. Do not turn off PRO restricted status without the consent of the system administrator or database administrator because updates to the partition might cause data loss.

**Table 143. Resetting PRO restricted status**

<table>
<thead>
<tr>
<th>Status</th>
<th>Abbreviation</th>
<th>Object affected</th>
<th>Corrective action</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Persistent Read Only</td>
<td>PRO</td>
<td>Table space partition</td>
<td>None</td>
<td>1, 2</td>
</tr>
</tbody>
</table>

**Note:**
1. MODIFY RECOVERY TABLESPACE DSNUM(n) forces the retention of the last two full image copies.
2. COPY TABLESPACE DSNUM(n) does not create a new full image copy of the partition.

### REBUILD-pending status

A REBUILD-pending restrictive status indicates that the affected index or index partition is broken and must be rebuilt from the data.

REBUILD-pending (RBDP) status indicates that the physical or logical partition is inaccessible and must be rebuilt. RBDP status is set on a data-partitioned secondary index if you create the index after performing the following actions:

- Create a partitioned table space.
- Create a partitioning index.
- Insert a row into a table.

In this situation, the last partition of the table space is set to REORG-pending (REORP) restrictive status.

REBUILD-pending star (RBDP*) status indicates that a logical partition of a nonpartitioned secondary index is unavailable for read-write access and the entire index is unavailable for read access.

Page set REBUILD-pending (PSRBD) status indicates that an entire nonpartitioned secondary index or index on the auxiliary table is unavailable for read-write access.

Rebuilding an index and thereby resetting the REBUILD-pending status invalidates the dynamic statement cache for the related table.

If you alter the data type of a column to a numeric data type, RECOVER INDEX cannot complete. You must rebuild the index.
Refer to the following table for information about resetting a REBUILD-pending status. This table lists the status name, abbreviation, affected objects, and any corrective actions.

<table>
<thead>
<tr>
<th>Status</th>
<th>Abbreviation</th>
<th>Object affected</th>
<th>Corrective action</th>
</tr>
</thead>
<tbody>
<tr>
<td>REBUILD-pending</td>
<td>RBDP</td>
<td>Physical or logical index partition</td>
<td>Run the REBUILD utility on the affected index partition.</td>
</tr>
<tr>
<td>REBUILD-pending star</td>
<td>RBDP*</td>
<td>Logical partitions of nonpartitioned secondary indexes</td>
<td>Run REBUILD INDEX PART or RECOVER utility on the affected logical partitions.</td>
</tr>
<tr>
<td>Page set</td>
<td>PSRBD</td>
<td>Nonpartitioned secondary index, index on the auxiliary table</td>
<td>Run REBUILD INDEX ALL, the RECOVER utility, or run REBUILD INDEX listing all indexes in the affected index space.</td>
</tr>
</tbody>
</table>

The following actions also reset the REBUILD-pending status:
- Use LOAD REPLACE for the table space or partition.
- Use REPAIR SET INDEX with NORBDPEND on the index partition. Be aware that this does not correct the data inconsistency in the index partition. Use CHECK INDEX instead of REPAIR to verify referential integrity constraints.
- Start the database that contains the index space with ACCESS FORCE. Be aware that this does not correct the data inconsistency in the index partition.
- Run REORG INDEX SORTDATA on the affected index.

RECOVER-pending status

The RECOVER-pending (RECP) restrictive status indicates that a table space or table space partition is broken and must be recovered.

Refer to the following table for information about resetting the RECOVER-pending status. This table lists the status name, abbreviation, affected objects, and any corrective actions.

<table>
<thead>
<tr>
<th>Status</th>
<th>Abbreviation</th>
<th>Object affected</th>
<th>Corrective action</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECOVER-pending</td>
<td>RECP</td>
<td>Table space</td>
<td>Run the RECOVER utility on the affected object.</td>
</tr>
<tr>
<td>RECOVER-pending</td>
<td>RECP</td>
<td>Table space partition</td>
<td>Recover the partition.</td>
</tr>
<tr>
<td>RECOVER-pending</td>
<td>RECP</td>
<td>Index on the auxiliary table</td>
<td>Correct the RECOVER-pending status by using one of the following utilities:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• REBUILD INDEX</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• RECOVER INDEX</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• REORG INDEX SORTDATA</td>
</tr>
</tbody>
</table>
Table 145. Resetting RECOVER-pending status (continued)

<table>
<thead>
<tr>
<th>Status</th>
<th>Abbreviation</th>
<th>Object affected</th>
<th>Corrective action</th>
<th>Notes</th>
</tr>
</thead>
</table>
| RECOVER-pending | RECP         | Index space    | Run one of the following utilities on the affected index space to reset RECP, RBDP, RBDP*, or PSRBDP status:  
|               |              |                | • REBUILD INDEX                                                                 |       |
|               |              |                | • RECOVER INDEX                                                                 |       |
|               |              |                | • REORG INDEX SORTDATA                                                           |       |
| RECOVER-pending | RECP         | Any            | The following actions also reset the RECOVER-pending status:  
|               |              |                | • Use LOAD REPLACE for the table space or partition.                            |       |
|               |              |                | • Use REPAIR SET TABLESPACE or INDEX with NORCVRPEND on the table space or partition. Be aware that this does not correct the data inconsistency in the table space or partition. |       |
|               |              |                | • Start the database that contains the table space or index space with ACCESS FORCE. Be aware that this does not correct the data inconsistency in the table space or partition. |       |

REFRESH-pending status

When Db2 marks an object in refresh-pending (REFP) status, it also puts the object in RECOVER-pending (RECP) or REBUILD-pending (RBDP or PSRBD) status.

For a table space, run LOAD REPLACE or the RECOVER utility to resolve this restricted status, or use the REBUILD INDEX utility to rebuild the index from the data.

Related reference:
- “RECOVER-pending status” on page 1044
- “REBUILD-pending status” on page 1043
- “Syntax and options of the LOAD control statement” on page 242
- Chapter 26, “RECOVER,” on page 459
- Chapter 25, “REBUILD INDEX,” on page 427

REORG-pending status

The following three REORG-pending states indicate that an object either must or should be reorganized:

**REORG-pending (REORP) restrictive status**
Indicates that the object must be reorganized to apply definition changes before the data is accessible. REORP status must be reset for all adjacent logical partitions that are in REORP status.

**REORG-pending (AREO*) advisory status**
Indicates that the object needs to be reorganized for optimal performance. AREO* status can be reset on a single partition, regardless of whether adjacent partitions are also in AREO* status. The affected objects are not restricted and can be accessed by both readers and writers.
REORG-pending (AREOR) advisory status:
Indicates that the object should be reorganized to apply pending definition changes. AREOR status must be reset for all adjacent logical partitions that are in AREOR status. The affected objects are not restricted and can be accessed by both readers and writers. The affected objects are not modified until the REORG job that materializes the definition changes is run.

The following table lists the actions to take to reset one of these REORG-pending states.

Table 146. Resetting REORG-pending status

<table>
<thead>
<tr>
<th>Status</th>
<th>Abbreviation</th>
<th>Object affected</th>
<th>Corrective action</th>
</tr>
</thead>
<tbody>
<tr>
<td>REORG-pending</td>
<td>REORP</td>
<td>Table space or partition</td>
<td>Run one of the following utilities on the table space or one or more partitions:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• REORG TABLESPACE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• LOAD with the REPLACE option</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If a table space is in both REORG-pending and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CHECK-pending status (or auxiliary</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CHECK-pending status), run REORG first and then</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>run the CHECK DATA utility to clear the respective</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>states.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If one or more partitions are in REORP status, you</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>must include all adjacent logical partitions that are in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>REORP status in the same REORG or LOAD job. For</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>example, suppose that a partitioned table space has</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>logical partitions in the following states:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P1  REORP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P2  REORP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P3  RW</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P4  REORP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P5  REORP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>In this case, the partition range P1:P2 must be in the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same utility job, and the partition range P4:P5 must</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>be in the same utility job. You can either include both</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>of these ranges in one utility job or run two separate</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>jobs, one for each range. Alternatively, you can</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>specify the SCOPE PENDING option so that REORG</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>includes both ranges in REORP status.</td>
</tr>
<tr>
<td>Advisory</td>
<td>AREO*</td>
<td>Table space or partition</td>
<td>Run one of the following utilities on the table space or one or more partitions:</td>
</tr>
<tr>
<td>REORG-pending</td>
<td></td>
<td></td>
<td>• REORG TABLESPACE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• LOAD with the REPLACE option</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If one or more partitions are in AREO* status, you</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>can reset AREO* for a specific partition without</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>being restricted by another AREO* status for an</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>adjacent partition. AREO* status does not have to be</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>reset for all affected partitions at the same time.</td>
</tr>
</tbody>
</table>
### Table 146. Resetting REORG-pending status (continued)

<table>
<thead>
<tr>
<th>Status</th>
<th>Abbreviation</th>
<th>Object affected</th>
<th>Corrective action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advisory REORG-pending</td>
<td>AREO*</td>
<td>Index space or partition</td>
<td>Run one of the following utilities on the index space or one or more partitions:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• REORG TABLESPACE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• LOAD with the REPLACE option</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• REORG INDEX</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• REBUILD INDEX</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If one or more partitions are in AREO* status, you can reset AREO* for a specific partition without being restricted by another AREO* status for an adjacent partition. AREO* status does not have to be reset for all affected partitions at the same time.</td>
</tr>
<tr>
<td>Advisory REORG-pending</td>
<td>AREOR</td>
<td>Table space or partition</td>
<td>Run REORG TABLESPACE on the table space or all affected partitions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>When AREOR is set at the partition level, you must include all adjacent logical partitions that are in REORF status in the same REORG job. For example, suppose that you change partition limits or change the page size and as a result, the table space has partitions in the following states:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P1 AREOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P2 AREOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P3 RW</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P4 AREOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P5 AREOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>In this case, the partition range P1:P2 must be in the same REORG job, and the partition range P4:P5 must be in the same REORG job. You can either include both of these ranges in one REORG job or run two separate jobs, one for each range. Alternatively, you can specify the SCOPE PENDING option so that REORG includes both ranges in AREOR status. (You can run REORG on the individual partitions and not the entire adjacent range in AREOR. However, in that case, REORG does not materialize any pending definition changes and the AREOR status is not reset.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If you want to drop any pending definition changes, use the ALTER TABLESPACE statement with the DROP PENDING CHANGES clause. This statement removes the pending changes and resets the AREOR status.</td>
</tr>
</tbody>
</table>
### Table 146. Resetting REORG-pending status (continued)

<table>
<thead>
<tr>
<th>Status</th>
<th>Abbreviation</th>
<th>Object affected</th>
<th>Corrective action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advisory</td>
<td>AREOR</td>
<td>Index space or partition</td>
<td>Run one of the following utilities on the index space or all affected partitions:</td>
</tr>
<tr>
<td>REORG-pending</td>
<td></td>
<td></td>
<td>• REORG TABLESPACE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• REORG INDEX</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>When AREOR is set at the partition level, you must include all adjacent logical partitions that are in AREOR status in the same REORG job. See the example in the preceding row for table spaces. (You can run REORG on the individual partitions and not the entire adjacent range in AREOR. However, in that case, REORG does not materialize any pending definition changes and the AREOR status is not reset.)</td>
</tr>
</tbody>
</table>

**Related reference:**

- ALTER TABLESPACE (Db2 SQL)
- Chapter 28, “REORG TABLESPACE,” on page 553
- Chapter 27, “REORG INDEX,” on page 515
- Chapter 29, “REPAIR,” on page 669
- Chapter 19, “LOAD,” on page 239

### Restart-pending status

If an object has backout work pending at the end of Db2 restart, the object is placed in restart-pending (RESTP) status.

You cannot use LOAD REPLACE on an object that is in the RESTP status.

Utility activity against page sets or partitions with RESTP status is not allowed. Any attempt to access a page set or partition with RESTP status terminates with return code 8.

The following table explains how to reset the restart-pending status.

### Table 147. Resetting restart-pending status

<table>
<thead>
<tr>
<th>Status</th>
<th>Abbreviation</th>
<th>Object affected</th>
<th>Corrective action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restart-</td>
<td>RESTP</td>
<td>Table space, table space partitions, index spaces,</td>
<td>Delay running REORG TABLESPACE SHRLEVEL CHANGE until all RESTP statuses are reset.</td>
</tr>
<tr>
<td>pending</td>
<td></td>
<td>and physical index space partitions</td>
<td></td>
</tr>
</tbody>
</table>

**Related tasks:**

---

1048  Utility Guide and Reference
Starting a table space or index space that has restrictions (Db2 Administration Guide)

Related reference:

- LIMIT BACKOUT field (LBACKOUT subsystem parameter) (Db2 Installation and Migration)
- RECOVER POSTPONED (Db2) (Db2 Commands)
Appendix C. DSN1SMFP

The DSN1SMFP utility processes Db2 trace data into reports that are useful for evaluating and auditing the evaluated configuration.

**PSPI** DSN1SMFP accepts data that SMF collects in standard SMF format and produces from one to eighteen reports. DSN1SMFP accepts all SMF record types, but it processes only type 101 (Db2 Accounting) and 102 (Db2 Performance) records. DSN1SMFP checks each type 101 and 102 record for Db2 audit trace types of these Db2 trace records:

- 003: Accounting - DDF Data by Location (security-relevant fields only)
- 004: Trace Start
- 005: Trace Stop
- 023: Utility Start
- 024: Utility Change
- 025: Utility End
- 083: Identify End
- 106: System Parameters (security-relevant fields only)
- 140: Audit Authorization Failures
- 141: Audit DDL Grant/Revoke
- 142: Audit DDL Create/Alter/Drop
- 143: Audit First Write
- 144: Audit First Read
- 145: Audit DML Statement
- 269: Trusted Connection
- 270: Trusted Context
- 350: SQL Statement
- 361: Audit Administrative Authorities
- 362: Trace Start and Stop with AUDITPOLICY.

IFCIDs: Each such trace type is extracted and outputted in report format to a dedicated DD for that trace type.

**Important:** At a minimum, ensure that the following traces are started when you collect SMF records for input to DSN1SMFP.

Table 148. Traces to start for DSN1SMFP

<table>
<thead>
<tr>
<th>Trace type</th>
<th>Class</th>
<th>IFCID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accounting</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Accounting</td>
<td>1</td>
<td>106</td>
</tr>
<tr>
<td>Audit</td>
<td>1</td>
<td>140</td>
</tr>
<tr>
<td>Audit</td>
<td>2</td>
<td>141</td>
</tr>
<tr>
<td>Audit</td>
<td>3</td>
<td>142</td>
</tr>
<tr>
<td>Audit</td>
<td>4</td>
<td>143</td>
</tr>
<tr>
<td>Audit</td>
<td>5</td>
<td>144</td>
</tr>
<tr>
<td>Audit</td>
<td>6</td>
<td>145</td>
</tr>
<tr>
<td>Audit</td>
<td>7</td>
<td>83</td>
</tr>
<tr>
<td>Audit</td>
<td>8</td>
<td>23</td>
</tr>
</tbody>
</table>
Table 148. Traces to start for DSN1SMFP (continued)

<table>
<thead>
<tr>
<th>Trace type</th>
<th>Class</th>
<th>IFCID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audit</td>
<td>8</td>
<td>24</td>
</tr>
<tr>
<td>Audit</td>
<td>8</td>
<td>25</td>
</tr>
<tr>
<td>Audit</td>
<td>10</td>
<td>269</td>
</tr>
<tr>
<td>Audit</td>
<td>10</td>
<td>270</td>
</tr>
<tr>
<td>Audit</td>
<td>11</td>
<td>361</td>
</tr>
<tr>
<td>Audit</td>
<td></td>
<td>362</td>
</tr>
<tr>
<td>Performance</td>
<td>30</td>
<td>350</td>
</tr>
</tbody>
</table>

IFCID 4 and 5 are started automatically when you start or stop any other trace, and IFCID 362 is started automatically when you start an audit policy trace.

DSN1SMFP also counts and reports the following values in the end-of-job summary:

- Total SMF records that are read
- Total SMF type 101 records
- Total SMF type 102 records
- Separate totals for IFCID 0003, 0004, 0005, 0023, 0024, 0025, 0083, 0106, 0140, 0141, 0142, 0143, 0144, 0145, 0269, 0270, 0350, 0361, 0362, and other IFCID records that are read
- Separate totals for formatted IFCID 0003, 0004, 0005, 0023, 0024, 0025, 0083, 0106, 0140, 0141, 0142, 0143, 0144, 0145, 0269, 0270, 0350, 0361, 0362 records that are written
- Total formatted records that are written

You can allocate each output DD of interest to a print device, data set, or held output.

DSN1SMFP loads the DSNHDECP module to obtain the EBCDIC CCSID for conversion of Unicode-encoded strings. You need to ensure that the library where the local DSNHDECP module resides (typically prefix.SDNSEXIT) is available and allocated ahead of prefix.SDNLLOAD in the JOBLIB or STEPLIB DD concatenation.

When DSN1SMFP encounters an IFCID 0106 trace record, if the EBCDIC CCSID that is used by Db2 does not agree with the CCSID that is loaded from DSNHDECP, DSN1SMFP reports a warning message and ends with a minimum return code of 4.

Related tasks:
- “Running DSN1SMFP” on page 1053

Related reference:
- “Before running DSN1SMFP” on page 1053
- “Sample DSN1SMFP control statement” on page 1054
- “DSN1SMFP output” on page 1054
Before running DSN1SMFP

You need to know the required environment, authorization, and control statement of the DSN1SMFP utility before you run it.

Environment

DSN1SMFP runs as a batch z/OS job. Db2 does not need to be started.

Required authorization

DSN1SMFP does not require authorization. However, if any of the SMF records are RACF-protected, the authorization ID of the job must have READ RACF authority.

Control statement

See “Sample DSN1SMFP control statement” on page 1054 for an example of using DSN1SMFP to extract, format, and print the SMF records.

Required input

- Specify the SMF records to extract, format, and print in a data set that is specified by an SMFIN DD statement.
- Include the data set that contains the DSNHDECP load module for the subsystem in the JOBLIB DD statement concatenation or the STEPLIB DD statement concatenation, before the prefix.SDSNLOAD data set.

Restriction: The DSN1SMFP utility cannot process compressed trace records. If your SMF trace records are compressed as a result of a YES setting on the SMFCOMP subsystem parameter, use the DSNTSMFD utility to decompress compressed trace records prior to running DSN1SMFP. If you attempt to run DSN1SMFP on compressed trace records, an abend occurs.

Running DSN1SMFP

About this task

Sample job DSNTEJCC provides the basic JCL framework that is needed to run DSN1SMFP. See the DSNTEJCC prolog for directions on how to customize it for use at your site.

Procedure

To run the DSN1SMFP utility:

Use the following EXEC statement:

```bash
//stepname EXEC PGM=DSN1SMFP
```
**Sample DSN1SMFP control statement**

The following statements specify that DSN1SMFP is to extract, format, and print the SMF records that are specified in the SMFIN DD statement.

```plaintext
//DSN1SMFP EXEC PGM=DSN1SMFP,COND=(4,LT)
//STPLIB DD DISP=SHR,DSN=prefix.SDSNEXIT <- req'd for local DSNHDECP
// DD DISP=SHR,DSN=prefix.SDSNLOAD
//SMFIN DD DISP=SHR,DSN=SMF records with DB2 trace data
//SYSPRINT DD SYSOUT=*
//IFCID003 DD ...
//IFCID004 DD ...
//IFCID005 DD ...
//IFCID006 DD ...
//IFCID024 DD ...
//IFCID025 DD ...
//IFCID083 DD ...
//IFCID106 DD ...
//IFCID140 DD ...
//IFCID141 DD ...
//IFCID142 DD ...
//IFCID143 DD ...
//IFCID144 DD ...
//IFCID145 DD ...
//IFCID269 DD ...
//IFCID270 DD ...
//IFCID350 DD ...
//IFCID361 DD ...
//IFCID362 DD ...
```

---

**DSN1SMFP output**

Use this utility to aid in determining and correcting system security problems. When diagnosing Db2, you might need to refer to licensed documentation to interpret output from this utility.

DSN1SMFP generates up to nineteen types of reports, according to the input, plus an end-of-job summary report. Each type of report is triggered by a particular IFCID type. Each occurrence of a recognized IFCID type causes a new report to be appended to any earlier report that was generated in the same run. Each report begins with a standard report header. When all records have been processed, an end of job report summarizes the input that was received, which records were processed, and any error messages that were encountered along the way.

DSN1SMFP creates the following dedicated report DDs:

### Table 149. DSN1SMFP dedicated report DDs

<table>
<thead>
<tr>
<th>DD name</th>
<th>Description</th>
<th>Record format</th>
<th>Record structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFCID003 DD</td>
<td>Extracted IFCID 003 (DDF Location) records</td>
<td>RECFM=FBA</td>
<td>See “IFCID003: DDF Location” on page 1057</td>
</tr>
<tr>
<td>IFCID004 DD</td>
<td>Extracted IFCID 004 (Trace Start) records</td>
<td>RECFM=FBA</td>
<td>See “IFCID004: Trace Start” on page 1059</td>
</tr>
</tbody>
</table>
### Table 149. DSN1SMFP dedicated report DDs (continued)

<table>
<thead>
<tr>
<th>DD name</th>
<th>Description</th>
<th>Record format</th>
<th>Record structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFCID005 DD</td>
<td>Extracted IFCID 005 (Trace Stop) records</td>
<td>RECFM=FBA</td>
<td>See “IFCID005: Trace Stop” on page 1059</td>
</tr>
<tr>
<td>IFCID023 DD</td>
<td>Extracted IFCID 023 (Utility Start) records</td>
<td>RECFM=FBA</td>
<td>See “IFCID023: Utility Start” on page 1059</td>
</tr>
<tr>
<td>IFCID024 DD</td>
<td>Extracted IFCID 024 (Utility Change) records</td>
<td>RECFM=FBA</td>
<td>See “IFCID024: Utility Change” on page 1063</td>
</tr>
<tr>
<td>IFCID025 DD</td>
<td>Extracted IFCID 025 (Utility End) records</td>
<td>RECFM=FBA</td>
<td>See “IFCID025: Utility End” on page 1064</td>
</tr>
<tr>
<td>IFCID083 DD</td>
<td>Extracted IFCID 083 (Identify End) records</td>
<td>RECFM=FBA</td>
<td>See “IFCID083: Identify End” on page 1065</td>
</tr>
<tr>
<td>IFCID106 DD</td>
<td>Extracted IFCID 106 (System Parameters) records (security-relevant fields only)</td>
<td>RECFM=FBA</td>
<td>See “IFCID106: System Parameters (security-relevant fields only)” on page 1066</td>
</tr>
<tr>
<td>IFCID140 DD</td>
<td>Extracted IFCID 140 (Audit Authorization Failures) records</td>
<td>RECFM=FBA</td>
<td>See “IFCID140: Audit Authorization Failures” on page 1075</td>
</tr>
<tr>
<td>IFCID141 DD</td>
<td>Extracted IFCID 141 (Audit DDL Grant/Revoke) records</td>
<td>RECFM=FBA</td>
<td>See “IFCID141: Audit DDL Grant/Revoke” on page 1081</td>
</tr>
<tr>
<td>IFCID142 DD</td>
<td>Extracted IFCID 142 (Audit DDL Create/Alter/Drop) records</td>
<td>RECFM=FBA</td>
<td>See “IFCID142: Audit DDL Create/Alter/Drop” on page 1082</td>
</tr>
<tr>
<td>IFCID143 DD</td>
<td>Extracted IFCID 143 (Audit First Write) records</td>
<td>RECFM=FBA</td>
<td>See “IFCID143: Audit First Write” on page 1084</td>
</tr>
<tr>
<td>IFCID144 DD</td>
<td>Extracted IFCID 144 (Audit First Read) records</td>
<td>RECFM=FBA</td>
<td>See “IFCID144: Audit First Read” on page 1084</td>
</tr>
<tr>
<td>IFCID145 DD</td>
<td>Extracted IFCID 145 (Audit DML Statement) records</td>
<td>RECFM=FBA</td>
<td>See “IFCID145: Audit DML Statement” on page 1085</td>
</tr>
<tr>
<td>IFCID269 DD</td>
<td>Extracted IFCID 269 (Trusted Connection) records</td>
<td>RECFM=FBA</td>
<td>See “IFCID269: Trusted Connection” on page 1087</td>
</tr>
<tr>
<td>IFCID270 DD</td>
<td>Extracted IFCID 270 (Trusted Context) records</td>
<td>RECFM=FBA</td>
<td>See “IFCID270: Trusted Context” on page 1089</td>
</tr>
<tr>
<td>IFCID350 DD</td>
<td>Extracted IFCID 350 (SQL Statement) records</td>
<td>RECFM=FBA</td>
<td>See “IFCID350: SQL Statement” on page 1089</td>
</tr>
<tr>
<td>IFCID361 DD</td>
<td>Extracted IFCID 361 (Audit Administrative Authorities) records</td>
<td>RECFM=FBA</td>
<td>See “IFCID361: Audit Administrative Authorities” on page 1091</td>
</tr>
<tr>
<td>IFCID362 DD</td>
<td>Extracted IFCID 362 (Trace Start and Stop with AUDITPOLICY) records</td>
<td>RECFM=FBA</td>
<td>See “IFCID362: Trace Start and Stop with AUDITPOLICY” on page 1095</td>
</tr>
<tr>
<td>SYSPRINT DD</td>
<td>Runtime messages and EOJ summary</td>
<td>RECFM=FBA</td>
<td>See “SYSPRINT: Runtime messages and end-of-job summary” on page 1097</td>
</tr>
</tbody>
</table>

For details about the record structure of the DSN1SMFP output, see “DSN1SMFP output record structure” on page 1056.
DSN1SMFP output record structure

The DSN1SMFP utility extracts, formats, and prints SMF records to dedicated report records. Each trace type report record has a different structure.

Standard report header

Each report, except the end-of-job report, begins with the standard report header.

Fields

The standard report header has the following fields:

**PRIMAUTH**
- The authorization ID under which the transaction is running.
  - Field Name: QWHCAID

**ORIGAUTH**
- The original authorization ID under which the transaction started.
  - Field Name: QWHCOPID

**PLANNNAME**
- The Db2 plan name.
  - Field Name: QWHCPLAN

**CONNECT**
- The connection ID.
  - Field Name: QWHCCN

**CORRNAME**
- The correlation name.
  - Field Name: QWHCCV

**CORRNMBR**
- The correlation number.
  - Field Name: QWHCCV

**INSTANCE**
- The unique number that is assigned to a thread.
  - Field Name: QWHSLUUV

**CONNTYPE**
- The type of connection that is being used to interface with Db2.
  - Field Name: QWHATYP

**END_USER**
- The user ID of the workstation user.
  - Field Name: QWHCEUID

**RECORD TIME**
- The timestamp that is contained in the trace record. The format is hours, minutes, seconds, and hundred-millionths of a second.
  - Field Name: QWHSSTCK
TCB CPU TIME
The CPU time that is stored in the trace record. The format is minutes, seconds, and hundred-millionths of a second.

Field Name: QWHUCPU

WS_NAME
The name of the workstation.

Field Name: QWHCEUWN

DESTNO
The destination sequence number.

Field Name: QWHSWSEQ

IFCID
The instrumentation facility component identification (Db2 trace record type).

Field Name: QWHSIID

DESCRIPTION
A brief description of the IFCID record. The description indicates whether the record contains accounting, statistics, or performance data. For performance data, the description also indicates the event.

TRANSACT
The name of the workstation transaction.

Field Name: QWHCEUTX

Example

<table>
<thead>
<tr>
<th>PRIMAUTH</th>
<th>CONNECT</th>
<th>INSTANCE</th>
<th>END_USER</th>
<th>WS_NAME</th>
<th>IFC DESCRIPTION</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORIGAUTH</td>
<td>CORRNAME</td>
<td>CONNTYPE</td>
<td>RECORD_TIME</td>
<td>DESTNO</td>
<td>TCB CPU TIME</td>
<td>ID</td>
</tr>
<tr>
<td>PLANNAME</td>
<td>CORRINMBR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>--------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSADEM</td>
<td>BATCH</td>
<td>X'0EF0F0E22787'</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSADEM</td>
<td>CTRACE4</td>
<td>TSO</td>
<td>01:02:39</td>
<td>0000006030</td>
<td>003 DDF Data by Location</td>
<td></td>
</tr>
<tr>
<td>DSNBINP</td>
<td>'BLANK'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**IFCID003: DDF Location**

The IFCID003 DD contains the extracted IFCID 003 (DFD Location) records. Only IFCID 0003 records that contain a distributed data facility statistics (DSNDQDAC) block are reported; other IFCID 0003 records are ignored. Only selected IFCID 0003 fields are used.

**Fields**

The IFCID003 DD has the following fields:

**REMOTE LOCATION**
The name of the remote location that this information is associated with. If the local location is the requester, this field is a server location. If the local location is a server location, this field is the requester location. An allied thread is created at a Db2 requester, and a database access thread is created at a Db2 server. An accounting record is for either a requester or a server, but not for both.
Field Name: QLACLOCN

**COMMIT SENT**
The number of single-phase commit requests that are sent to the server location. This value is maintained at the requester location.

Field Name: QLACCOMS

**COMMIT RECEIVED**
The number of single-phase commit requests that are received from the requester location. This value is maintained at the server location.

Field Name: QLACOMR

**REQ.ELAPSED TIME**
The elapsed time at the requester location until the database access agent completed its work, including Db2 processing time and network time. This value is maintained at the requester location and is calculated by accumulating the difference between the store clock values that are obtained before and after each network request.

Field Name: QLACCPUL

**ROLLBK SENT**
The number of rollback requests that were sent to the server location (single-phase commit operations only). This value is maintained at the requester location.

Field Name: QLACABRS

**SER.ELAPSED TIME**
The elapsed database access agent time at the server location. This time is updated at the requester location and is reported only for Db2 private protocol. If the agent uses both methods to communicate with the server location, only the elapsed time that is associated with the Db2 private protocol is reported, and this time can be misleading. If only DRDA is used, this value is 0.

Field Name: QLACCPUR

**ROLLBK RECEIVED**
The number of rollback requests that are received from the requester location (single-phase commit operations only). This value is maintained at the server location.

Field Name: QLACABRR

### Example

<table>
<thead>
<tr>
<th>DDF DATA BY LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTE LOCATION</td>
</tr>
<tr>
<td>COMMIT SENT</td>
</tr>
<tr>
<td>COMMIT RECEIVED</td>
</tr>
<tr>
<td>REMOTE LOCATION</td>
</tr>
<tr>
<td>ROLLBK SENT</td>
</tr>
<tr>
<td>ROLLBK RECEIVED</td>
</tr>
<tr>
<td>---------------------</td>
</tr>
<tr>
<td>STLEC1B</td>
</tr>
<tr>
<td>0000000000</td>
</tr>
<tr>
<td>000000000000</td>
</tr>
<tr>
<td>09:48:00</td>
</tr>
<tr>
<td>0000000000</td>
</tr>
<tr>
<td>09:48:00</td>
</tr>
<tr>
<td>0000000000</td>
</tr>
</tbody>
</table>
IFCID004: Trace Start

The IFCID004 DD contains the extracted IFCID 004 (Trace Start) records.

Fields

MESSAGE
The start trace message.
Field Name: QW0004MS
QW0004CM
For serviceability only.
Field Name: QW0004CM

Example
MESSAGE: -START TRACE (AUDIT AUDPLCY )CLASS (*)RMID (*)DEST (SMF )PLAN (*)AUTHID (*)IFCID (*)BUFSIZE (*)USERID (*)APPNAME (*)WRSKTN (*)PKGLOC (*)PKGCOL (*)PKGPROG (*)CONNID (*)CORRID (*)ROLE (*)..
QW0004CM: X'60E2E3C1D9E340E3'

IFCID005: Trace Stop

The IFCID005 DD contains the extracted IFCID 005 (Trace Stop) records.

Fields

MESSAGE
The stop trace message.
Field Name: QW0004MS
QW0005CM
For serviceability only.
Field Name: QW0005CM

Example
MESSAGE: -STOP TRACE (AUDIT )CLASS (*)RMID (*)PLAN (*)AUTHID (*)TND (*)USERID (*)APPNAME (*)WRSKTN (*)PKGLOC (*)PKGCOL (*)PKGPROG (*)CONNID (*)CORRID (*)ROLE (*)..
QW0005CM: X'60E2E3D6'

IFCID023: Utility Start

The IFCID023 DD contains the extracted IFCID 023 (Utility Start) records.
Fields

The IFCID023 DD has the following fields:

**DBID**  The database ID.
  Field Name: QW0023DB

**UTILITY NAME**  The utility name.
  Field Name: QW0023NM

**OBID**  The object ID.
  Field Name: QW0023PD

**UTILITY PHASE**  The phase name of the utility.
  Field Name: QW0023PH

**RQSTASK**  The number of requested subtasks.
  Field Name: QW0023R1

**UTILITY ID**  The identifier of the utility.
  Field Name: QW0023ID

**REORG KEEPDICTIONARY**  The REORG utility is specified with the KEEPDICTIONARY keyword.
  Field Name: QW0023D1

**LOAD KEEPDICTIONARY**  The LOAD utility is specified with the KEEPDICTIONARY keyword.
  Field Name: QW0023B1

**COPY CONCURRENT**  The COPY utility is specified with the CONCURRENT keyword.
  Field Name: QW0023A1

**REBUILD REUSE**  The REBUILD utility is specified with the REUSE keyword.
  Field Name: QW0023F1

**REORG REUSE**  The REORG utility is specified with the REUSE keyword.
  Field Name: QW0023D2

**LOAD REUSE**  The LOAD utility is specified with the REUSE keyword.
  Field Name: QW0023B2

**COPY SHRLEVEL CHANGE**  The COPY utility is specified with the SHRLEVEL(CHANGE) keyword.
  Field Name: QW0023A2
REBUILD SORTKEYS
The REBUILD utility is specified with the SORTKEYS keyword.
Field Name: QW0023F2

REORG LOG NO
The REORG utility is specified with the LOG(NO) keyword.
Field Name: QW0023D3

LOAD LOG NO
The LOAD utility is specified with the LOG(NO) keyword.
Field Name: QW0023B3

COPY PARALLEL
The COPY utility is specified with the PARALLEL keyword.
Field Name: QW0023A3

REBUILD STATISTICS
The REBUILD utility is specified with the STATISTICS keyword.
Field Name: QW0023F3

REORG SORTKEYS
The REORG utility is specified with the SORTKEYS keyword.
Field Name: QW0023D4

LOAD SORTKEYS
The LOAD utility is specified with the SORTKEYS keyword.
Field Name: QW0023B4

COPY CHECKPAGE
The COPY utility is specified with the CHECKPAGE keyword.
Field Name: QW0023A4

REBUILD WORKDDN
The REBUILD utility is specified with the WORKDDN keyword.
Field Name: QW0023F4

REORG SORTDATA
The REORG utility is specified with the SORTDATA keyword.
Field Name: QW0023D5

LOAD SHRLEVEL CHANGE
The LOAD utility is specified with the SHRLEVEL(CHANGE) keyword.
Field Name: QW0023B5

REORG NOSYSREC
The REORG utility is specified with the NOSYSREC keyword.
Field Name: QW0023D6

LOAD COPYDDN
The LOAD utility is specified with the COPYDDN keyword.
Field Name: QW0023B6

RECOVER REUSE
The RECOVER utility is specified with the REUSE keyword.
Field Name: QW0023A7
RUNSTATS SAMPLE
The RUNSTATS utility is specified with the SAMPLE keyword.
Field Name: QW0023G1

REORG SHRLEVEL CHANGE
The REORG utility is specified with the SHRLEVEL(CHANGE) keyword.
Field Name: QW0023D7

LOAD STATISTICS
The LOAD utility is specified with the STATISTICS keyword.
Field Name: QW0023B7

RECOVER PARALLEL
The RECOVER utility is specified with the PARALLEL keyword.
Field Name: QW0023A8

RUNSTATS SHRLEVEL CHANGE
The RUNSTATS utility is specified with the SHRLEVEL(CHANGE) keyword.
Field Name: QW0023G2

REORG SHRLEVEL REFERENCE
The REORG utility is specified with the SHRLEVEL(REFERENCE) keyword.
Field Name: QW0023B8

LOAD PART INDDN
The LOAD utility is specified with the PART(INDDN) keyword.
Field Name: QW0023B8

REORG COPYDDN
The REORG utility is specified with the COPYDDN keyword.
Field Name: QW0023E1

UNLOAD SHRLEVEL REFERENCE
The UNLOAD utility is specified with the SHRLEVEL(REFERENCE) keyword.
Field Name: QW0023H1

REORG STATISTICS
The REORG utility is specified with the STATISTICS keyword.
Field Name: QW0023E2

UNLOAD SHRLEVEL CHANGE ISOLATION CS
The UNLOAD utility is specified with the SHRLEVEL(CHANGE ISOLATION CS) keyword.
Field Name: QW0023H2

REORG FASTSWITCH
The REORG utility is specified with the FASTSWITCH keyword.
Field Name: QW0023E3

UNLOAD SHRLEVEL CHANGE ISOLATION UR
The UNLOAD utility is specified with the SHRLEVEL(CHANGE ISOLATION UR) keyword.
**Example**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBID</td>
<td>Database ID.</td>
</tr>
<tr>
<td>UTILITY NAME</td>
<td>Utility name.</td>
</tr>
<tr>
<td>OBID</td>
<td>Object ID.</td>
</tr>
<tr>
<td>UTILITY PHASE</td>
<td>Phase name of the utility.</td>
</tr>
<tr>
<td>RQSTASK</td>
<td>Request task.</td>
</tr>
<tr>
<td>UTILITY ID</td>
<td>Identifier of the utility.</td>
</tr>
<tr>
<td>DBNAME</td>
<td>Database name.</td>
</tr>
<tr>
<td>OBJECT NAME</td>
<td>Table space name or index name.</td>
</tr>
</tbody>
</table>

**IFCID024: Utility Change**

The IFCID024 DD contains the extracted IFCID 024 (Utility Change) records.

**Fields**

The IFCID024 DD has the following fields:

- **DBID**  The database ID.
  - Field Name: QW0024DB

- **UTILITY NAME**  The utility name.
  - Field Name: QW0024NM

- **OBID**  The object ID.
  - Field Name: QW0024PD

- **UTILITY PHASE**  The phase name of the utility.
  - Field Name: QW0024PH

- **ITEMS**  The number of items processed by the utility.
  - Field Name: QW0024DN

- **UTILITY ID**  The identifier of the utility.
  - Field Name: QW0024ID

- **DBNAME**  The database name.
  - Field Name: QW0024NA

- **OBJECT NAME**  The table space name or index name.
Field Name: QW0024PN

PART/DATASET#

The number of the partition or data set if the utility is operating on one partition or data set. Otherwise, the value of this field is 0.

Field Name: QW0024PN

Example

| DBID     | 00269 | UTILITY NAME : REORG |
| OBID     | 00002 | UTILITY PHASE: UNLOAD |
| ITEMS    | 00000000000000000004 | UTILITY ID : NNCC1A3.REORG1 |
| DBNAME   | DBCCDB2A | OBJECT NAME : TSCCDB2A |
| PART/DATA SET# | 0000000000 |

IFCID025: Utility End

The IFCID025 DD contains the extracted IFCID 025 (Utility End) records.

Fields

The IFCID025 DD has the following fields:

DBID  The database ID.

Field Name: QW0025DB

UTILITY NAME  The utility name.

Field Name: QW0025NM

OBID  The object ID.

Field Name: QW0025PD

UTILITY PHASE  The phase name of the utility.

Field Name: QW0025PH

ITEMS  The number of items that are processed by the utility.

Field Name: QW0025DN

UTILITY ID  The identifier of the utility.

Field Name: QW0025ID

Example

| DBID     | 00269 | UTILITY NAME : REORG |
| OBID     | 00002 | UTILITY PHASE: |
| ITEMS    | 00000000000000000004 | UTILITY ID : NNCC1A3.REORG1 |
IFCID083: Identify End

The IFCID083 DD contains the extracted IFCID 083 (Identify End) records.

**Fields**

The IFCID083 DD has the following fields:

**RECOPT**
The record coordination option specification.
Field Name: QW0083RO

**ACCESS**
Whether the access was successful.
Field Name: QW0083AD

**CURR SQLID**
The current SQL authorization ID.
Field Name: QW0083QD

**ORIG AUTHID**
The original primary authorization ID.
Field Name: QW0083OP

**SECONDARY AUTHORIZATION IDS**
The list of the secondary authorization IDs. This list is only produced if there are secondary authorization IDs.
Field Name: QW0083SA

**ACEE UTOKEN**
The ACEE UTOKEN.
Field Name: QW0083UT

**RETURN CODE**
The return code.
Field Name: QW0083RT

**REASON CODE**
The reason code.
Field Name: QW0083RS

**CONNECTION TYPE**
The connection type.
Field Name: QW0083CT

**Example**

```
END
RECOPT: '....'
ACCESS: SUCCESSFUL
CURR SQLID: USRT005
ORIG AUTHID: USRT005
SECONDARY AUTHORIZATION IDS:
ACEE UTOKEN: &.....H.SYSHIGH STLV3JES2 STLV3SYS1 RDR1.......USRT005SYS1
RETURN CODE: 0000000000
REASON CODE: 0000000000
CONNECTION TYPE: X'C2C1E3C3C8404040'
```
IFCID106: System Parameters (security-relevant fields only)

The IFCID106 DD contains the extracted IFCID106 (System Parameters) records.

The IFCID106 DD has the following five sections for the security-relevant fields:
- System initialization parameters
- Miscellaneous installation parameters
- Distributed Data Facility (DDF) parameters
- Data sharing parameters
- Application programming defaults

Fields for system initialization parameters

The IFCID106 DD has the following fields for system initialization parameters:

**WTO ROUTE CODES**

The MVS console routing codes. These codes are assigned to messages that are not solicited from a specific console. Up to 16 comma-separated codes can be shown.

Install parameter: WTO ROUTE CODES on panel DSNTIPO
DSN6SYSP parameter: ROUTCDE
Field Name: QWP1SMRC

**MONITOR BUFFER SIZE**

The default number of bytes allocated for the monitor trace buffer.

Install parameter: MONITOR SIZE on panel DSNTIPN
DSN6SYSP parameter: MONSIZE
Field Name: QWP1MONS

**AUDIT CLASSES**

Shows whether the audit trace is started automatically when Db2 is started. When a value of YES is specified, the audit trace is started for the default class (class 1) whenever Db2 is started. When a value of ALL is specified, an audit trace is automatically started for all classes.

Install parameter: AUDIT TRACE on panel DSNTIPN
DSN6SYSP parameter: AUDITST
Field Name: QWP1AUDT

**EXT. SECURITY**

Extended security options. When the recommended value of YES is specified, detailed reason codes are returned to a DRDA level 3 client when a DDF connection request fails because of security errors. When using SNA protocols, the requester must have included a product that supports the extended security sense codes, such as Db2 Connect version 5 and subsequent releases. RACF users can change their passwords using the DRDA change password function. This support is only for DRDA level 3 requesters that have implemented support for changing passwords. A value of YES allows properly-enabled DRDA clients to determine the cause of security failures without requiring Db2 operator support. When a value
of NO is specified, generic error codes are returned to the clients and RACF users are prevented from changing their passwords.

Install parameter: EXTENDED SECURITY on panel DSNTIPR

DSN6SYSP parameter: EXTSEC

Field Name: QWP1SCER

**UNICODE IFCIDS**

Shows whether output from IFC records should contain character data in Unicode or EBCDIC. Only a subset of the character fields (identified in the IFCID record definition by a %U in the comment area to the right of the field declaration in the DSNDQWxx copy files) are encoded in Unicode. The remaining fields maintain the same encoding of previous releases.

Install parameter: UNICODE IFCIDS on panel DSNTIPN

DSN6SYSP parameter: UIFCIDS

Field Name: QWP1_UNICODE

**Example for system initialization parameters**

<table>
<thead>
<tr>
<th>SYSTEM INITIALIZATION PARAMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>WTO ROUTE CODES: X'8000' MONITOR BUFFER SIZE: 0001D4B576 AUDIT CLASSES: X'00000000' EXT. SECURITY: NO</td>
</tr>
<tr>
<td>DATABASE PROTOCOL: D UNICODE IFCIDS : NO</td>
</tr>
</tbody>
</table>

**Fields for miscellaneous installation parameters**

The IFCID106 DD has the following fields for miscellaneous installation parameters:

**COMCRIT**

Indicates whether the Db2 environment for Common Criteria is enabled.

Install parameter: None

DSN6SPRM parameter: COMCRIT

Field name: QWP4COMC

**DDL REGISTRATION FLAG**

The DDL registration facility flag. It summarizes four settings for data definition control support:

**Data definition control**

Indicates whether DDL statements are validated by data definition control support.

Install parameter: INSTALL DD CONTROL SUPT on panel DSNTIPZ

DSN6SPRM parameter: RGFINSTL

**Application control**

Indicates whether the Db2 system is controlled by a set of closed applications whose application identifiers are identified in the application registration table. Closed applications require their Db2 objects to be managed solely through the plans or packages of the closed application that is registered in the application registration table.

Install parameter: CONTROL ALL APPLICATIONS on panel DSNTIPZ
DSN6SPRM parameter: RGFDEDPL

**Full names required**
Indicates whether registered objects require fully-qualified names.
Install parameter: REQUIRE FULL NAMES on panel DSNTIPZ

DSN6SPRM parameter: RGFFULLQ

**Unregistered DDL action**
Indicates whether DDL that names an unregistered object is to be rejected, rejected if the current application is not registered, or accepted.

Install parameter: UNREGISTERED DDL DEFAULT on panel DSNTIPZ

DSN6SPRM parameter: RGFDEFLT

SDL REGISTRATION FLAG can have one of the following values:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Data definition control</th>
<th>Application control</th>
<th>Full names required</th>
<th>Unregistered DDL action</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Reject</td>
</tr>
<tr>
<td>08</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Reject</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>Reject</td>
</tr>
<tr>
<td>48</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Reject</td>
</tr>
<tr>
<td>68</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Reject</td>
</tr>
<tr>
<td>88</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Reject</td>
</tr>
<tr>
<td>A8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C0</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>Reject</td>
</tr>
<tr>
<td>C8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E0</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Reject</td>
</tr>
<tr>
<td>E8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Field Name: QWP4REGF
INSTALL SYSADM
One of two authorization IDs with SYSADM authority. SYSADM users can access to Db2 in all cases.

| Install parameter: SYSTEM ADMIN 1 on panel DSNTIPP1 |
| DSN6SPRM parameter: SYSADM |
| Field Name: QWP4SADM |

DEFAULT USERID
The authorization ID used if RACF is not available for batch access and USER= is not specified in the job statement.

| Install parameter: UNKNOWN AUTHID on panel DSNTIPP1 |
| DSN6SPRM parameter: DEFLTID |
| Field Name: QWP4DFID |

SYSADM ID 2
One of two authorization IDs with SYSADM authority. SYSADM users can access to Db2 in all cases.

| Install parameter: SYSTEM ADMIN 2 on panel DSNTIPP1 |
| DSN6SPRM parameter: SYSADM2 |
| Field Name: QWP4ADM2 |

SITE TYPE
Shows whether this system is at a local site or a recovery site.

LOCALSITE
This is the site of the current system. Multiple image copies are made and are operational here. This is the default.

RECOVERYSITE
This an alternative site for recovery purposes.

| Install parameter: SITE TYPE on panel DSNTIPO |
| DSN6SPRM parameter: SITETYP |
| Field Name: QWP4MSTY |

SYSOPER ID
One of two authorization IDs with SYSOPR authority. SYSOPR users can access Db2 even if the Db2 catalog is unavailable.

| Install parameter: SYSTEM OPERATOR 1 on panel DSNTIPP1 |
| DSN6SPRM parameter: SYSOPR1 |
| Field Name: QWP4OPR1 |

SYSOPER ID 2
One of two authorization IDs with SYSOPR authority. SYSOPR users can access Db2 even if the Db2 catalog is unavailable.

| Install parameter: SYSTEM OPERATOR 2 on panel DSNTIPP1 |
| DSN6SPRM parameter: SYSOPR2 |
| Field Name: QWP4OPR2 |

ENABLE Db2 AUTHORIZATION
Shows whether Db2 performs authorization checking. When all
authorization checking by Db2 is disabled, the GRANT statement is also disabled (granting every privilege to PUBLIC); this is not recommended.

Install parameter: USE PROTECTION on panel DSNTIPP

DSN6SPRM parameter: AUTH

Field Name: QWP4AUTH

**CACHE DYNAMIC SQL**

Indicates whether prepared dynamic use by eligible application processes.

Install parameter: CACHE DYNAMIC on panel DSNTIP8

DSN6SPRM parameter: CACHECDYN

Field Name: QWP4CDYN

**AUTH. CACHE SIZE**

The size of the authorization cache to use if no CACHESIZE is specified on the BIND PLAN subcommand. A value of 0 means authorization caching is not used.

Install parameter: PLAN AUTH CACHE on panel DSNTIPP

DSN6SPRM parameter: AUTHCACHE

Field Name: QWP4AUTHC

**PACK AUTH CACHE**

The amount of storage that is allocated for caching authorization information for all packages on this Db2 member.

Install parameter: PACKAGE AUTH CACHE on panel DSNTIPP

DSN6SPRM parameter: CACHEPAC

Field Name: QWP4PAC

**DBADM CREATE VIEW**

Shows whether a Db2 administrator can create a view or alias for another user. Possible values are YES or NO. The default value is NO.

Install parameter: DBADM CREATE AUTH on panel DSNTIPP

DSN6SPRM parameter: DBACRVIEW

Field Name: QWP4CRVIEW

**EDM STATEMNT CACHE**

The size of the statement cache that can be used by the Environmental Descriptor Manager (EDM).

Install parameter: EDM STATEMENT CACHE on panel DSNTIPC

DSN6SPRM parameter: EDMSTMTC

Field Name: QWP4ESTMTC

**ONL SYSPARM TYPE**

The type of Db2 system parameter that was changed by the last SET SYSPARM statement.

Install parameter: None

DSN6SPRM parameter: None

Field Name: QWP4OZTP
ONL SYSPARM USER ID
The user ID that made the last online change to Db2 system settings.
Install parameter: None
DSN6SPRM parameter: None
Field Name: QWP4OZUS

ONL SYSPARM CORID
The correlation ID of the online application that made the last change to
Db2 system settings.
Install parameter: None
DSN6SPRM parameter: None
Field Name: QWP4OZCI

ONL SYSPARM TIME
Time of the last online change made to Db2 system settings.
Install parameter: None
DSN6SPRM parameter: None
Field Name: QWP4OZTM

SECURITY ADMIN 1 TYPE
Specifies whether the entry in the SECURITY ADMIN 1 field is an
authorization ID or a role.
Install parameter: SEC ADMIN 1 TYPE field on panel DSNTIPP1
DSN6SPRM parameter: SECADM1_TYPE
Field Name: QWP4SECA1_Type

SECURITY ADMIN 2 TYPE
Specifies whether the entry in the SECURITY ADMIN 2 field is an
authorization ID or a role.
Install parameter: SEC ADMIN 2 TYPE field on panel DSNTIPP1
DSN6SPRM parameter: SECADM2_TYPE
Field Name: QWP4SECA2_Type

SECURITY TASKS
Specifies whether Db2 security administrator duties are to be separated
from system administrator duties for this subsystem. You must set the
SEPARATE_SECURITY system parameter to YES in the evaluated
configuration.
Install parameter: SEPARATE SECURITY field on panel DSNTIPP1
DSN6SPRM parameter: SEPARATE_SECURITY
Field Name: QWP4SEPSD

REVOKE DEP. PRIVILEGES
Specifies whether revoking a privilege from a user is to cause dependent
privileges to be revoked. If dependent privileges are to be revoked,
revoking a privilege from a user also revokes the privilege from anyone
that the user has granted that privilege to.
Install parameter: REVOKE DEP PRIV field on panel DSNTIPP1
DSN6SPRM parameter: REVOKE_DEP_PRIVILEGES
Field Name: QWP4RVDPR

**SECURITY ADMIN 1**

The first of two authorization IDs or roles that are assigned to have Db2 security administrator authority.

Install parameter: SECURITY ADMIN 1 on panel DSNTIPPI

DSN6SPRM parameter: SECADM1

Field Name: QWP4SECA1

**SECURITY ADMIN 2**

The second of two authorization IDs or roles that are assigned to have Db2 security administrator authority.

Install parameter: SECURITY ADMIN 2 on panel DSNTIPPI

DSN6SPRM parameter: SECADM2

Field Name: QWP4SECA2

**Example for miscellaneous installation parameters**

<table>
<thead>
<tr>
<th>COMMON CRITERIA ENVIRON</th>
<th>YES</th>
<th>DDL REGISTRATION FLAG: X‘30’</th>
<th>INSTALL SYSADM1</th>
<th>DEFAULT USERID</th>
<th>IBMUSER</th>
<th>SYSADM ID</th>
<th>SYSAADM2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENABLE DB2 AUTHORIZATION</td>
<td>YES</td>
<td>CACHE DYNAMIC SQL: NO</td>
<td>AUTH. CACHE SIZE: 01024</td>
<td>EDM STMNT CACHE: 0000000000 ONL SYSPARAM TYPE: N/A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PACK AUTH CACHE</td>
<td>0000000000</td>
<td>DBADM CREATE VIEW: YES</td>
<td>SECADM</td>
<td>SECURITY ADMIN 1 TYPE: AUTH ID SECURITY ADMIN 2 TYPE: DYNAMIC SQL AUTH. CACHE SIZE: 01024 HOP SITE AUTHORIZ.: YES</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ONL SYSPARAM CORID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SECURITY ADMIN 1</td>
<td>SECADM</td>
<td>SECURITY ADMIN 2</td>
<td>SECADM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Miscellaneous installation parameters**

**Fields for DDF parameters**

The IFCID106 DD has the following fields for DDF parameters:

**FACILITY NAME**

- The name of the DDF facility.

  Install parameter: None

  DSN6FAC parameter: None

  Field Name: QWP9NAME

**RESYNCH.INTERVAL**

- The number of minutes between resynchronization periods.

  Install parameter: RESYNCH INTERVAL on panel DSNTIPRI

  DSN6FAC parameter: RESYNC

  Field Name: QWP9RYS

**TCP/IP VERIFIED**

- Indicates whether Db2 accepts TCP/IP connection requests containing only a user ID.

  Install parameter TCP/IP ALREADY VERIFIED on panel DSNTIP5

  DSN6FAC parameter: TCPALVER

  Field Name: QWP9TCP
FACILITY START
Indicates whether DDF is loaded, and if so, how it was started.
Install parameter DDF STARTUP OPTION on panel DSNTIPR
DSN6FAC parameter: DDF
Field Name: QWP9STRT

DBAT STATUS
Shows whether Db2 inactivates threads that have successfully committed or rolled back, and hold no cursors.
Install parameter DDF THREADS on panel DSNTIPR
DSN6FAC parameter: CMTSTAT
Field Name: QWP9CMST

TCP/IP KEEPALIVE
Indicates whether the TCP/IP configuration KeepAlive value has been overwritten.
Install parameter TCP/IP KEEPALIVE on panel DSNTIP5
DSN6FAC parameter: TCPKPALV
Field Name: QWP9TCKA

Example for DDF parameters

| DISTRIBUTED DATA FACILITY PARAMETERS |
| FACILITY NAME: DDF | RESYNCH INTERVAL: 00002 | TCP/IP VERIFIED: NO | TCP/IP KEEPALIVE: ENABLE |
| DBAT STATUS: INACTIVE | FACILITY START: AUTO |

Fields for data sharing parameters

The IFCID106 DD has the following fields for data sharing parameters:

GROUP NAME
The name of the Db2 data-sharing group. A value of N/A indicates that this Db2 is not part of a data-sharing group.
Install parameter: GROUP NAME on panel DSNTIPK
DSN6GRP parameter: GRPNAME
Field Name: QWPAGRP

MEMBER NAME
The member name of this Db2. A value of N/A indicates that this Db2 is not part of a data-sharing group.
Install parameter: MEMBER NAME on panel DSNTIPK
DSN6GRP parameter: MEMBNAME
Field Name: QWPARMBR

DATA SHARING ENABLED
Indicates whether data sharing is enabled.
Install parameter: DATA SHARING on panel DSNTIP0A
DSN6GRP parameter: DSHARE
Field Name: QWPADSHR
PAR.COORD
Shows whether this Db2 member can coordinate parallel processing on other members of the group. A value of N/A indicates that this Db2 is not part of a data-sharing group.
Install parameter: COORDINATOR on panel DSNTIPK
DSN6GRP parameter: COORDNTR
Field Name: QWPACOOR

PAR.ASSIST
Shows whether this Db2 member can assist a parallelism coordinator with parallel processing. A value of N/A indicates that this Db2 is not part of a data-sharing group.
Install parameter: ASSISTANT on panel DSNTIPK
DSN6GRP parameter: ASSIST
Field Name: QWAASST

Example for data sharing parameters

<table>
<thead>
<tr>
<th>DATA SHARING PARAMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>GROUP NAME: DSNCAT</td>
</tr>
<tr>
<td>MEMBER NAME: DB2A</td>
</tr>
<tr>
<td>DATA SHARING ENABLED: NO</td>
</tr>
<tr>
<td>PAR.COORD: NO</td>
</tr>
<tr>
<td>PAR.ASSIST: NO</td>
</tr>
</tbody>
</table>

Fields for application programming defaults

The IFCID106 DD has the following fields for application programming defaults:

VERSION
The version, release, and modification level.
Install parameter: None
DSNHDECP parameter: None
Field Name: QWPBREL

DEFAULT SUBSYSTEM
The MVS subsystem name for Db2.
Install parameter: SUBSYSTEM NAME on panel DSNTIPM
DSNHDECP parameter: SSID
Field Name: QWPBSSID

EBCDIC SBCS CCSID
The EBCDIC single-byte coded character set ID.
Install parameter: EBCDIC CCSID on panel DSNTIPF
DSNHDECP parameter: SCCSID
Field Name: QWPBSID

DECIMAL POINT OPTION
Indicates whether the decimal contains a comma (,) or a period (.)
Install parameter: DECIMAL POINT IS on panel DSNTIPF
DSNHDECP parameter: DECIMAL
Field Name: QWPBDE
DEFAULT ENCODING SCHEME
The default encoding scheme, which can be ASCII, EBCDIC, or UNICODE.
Install parameter: DEF ENCODING SCHEME on panel DSNTIPF
DSNHDECP parameter: ENSCHEME
Field Name: QWPBENS

Example for application programming defaults

<table>
<thead>
<tr>
<th>APPLICATION PROGRAMMING DEFAULTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>VERSION : 1110</td>
</tr>
<tr>
<td>DEFAULT SUBSYSTEM : DB2A</td>
</tr>
<tr>
<td>EBCDIC SBCS CCSID: 00037</td>
</tr>
<tr>
<td>DECIMAL POINT OPTION: PERIOD</td>
</tr>
<tr>
<td>DEFAULT ENCODING SCHEME: EBCDIC</td>
</tr>
</tbody>
</table>

IFCID140: Audit Authorization Failures
The IFCID140 DD contains the extracted IFCID 140 (Audit Authorization Failures) records.

Fields

The IFCID140 DD has the following fields:

AUTH CHECKED
The authorization ID that is being checked.
Field Name: QW0140UR

AUTHID TYPE
The authorization ID type.
Field Name: QW0140AT

REASON
The user-defined reason code from the access control authorization exit routine.
Field Name: QW0140RS

STATMNT LENGTH
The length of the failing SQL statement plus 4. It has a value of zero if no SQL statement exists.
Field Name: QW0140LL

RETCOD
The return code from the access control authorization exit routine.
Field Name: QW0140RC

PRIV CHECKED
The privilege that is being checked. It can have the following values:
• ALL ON PACKAGES
• ALLPKAUT
• ALTER
• ALTER BUFFERPOOL
• ALTER INDEX
• ALTERIN
• ARCHIVE
• BIND ADD
• BIND, REBIND OR FREE
• BINDAGENT
• CHECK DATA UTILITY
• CHECK UTILITY
• COMMENT ON
• COMMENT ON INDEX
• COPY
• COPY PACKAGE
• CREATE ALIAS
• CREATE GLOBAL TEMPORARY TABLE (CREATETMTAB)
• CREATE IN
• CREATE INDEX
• CREATE SECURE OBJECT
• CREATE STOGROUP
• CREATE SYNONYM
• CREATE TABLE
• CREATE TABLESPACE
• CREATE VIEW
• CREATEDBA
• CREATEDBC
• CREATEIN
• DBADM
• DBCTRL
• DBMAINT
• DDF COMMAND - START, STOP, OR CANCEL
• DEBUG SESSION
• DELETE
• DESCRIBE TABLE
• DIAGNOSE UTILITY
• DISPLAY
• DISPLAY ARCHIVE
• DISPLAY BUFFERPOOL
• DISPLAY DATABASE
• DISPLAY PROFILE
• DISPLAY RLIMIT
• DISPLAY THREAD OR DISPLAY DB
• DISPLAY UTILITY
• DROP
• DROP ALIAS
• DROP INDEX
• DROP SYNONYM
• DROPIN
• EXECUTE
• EXPLAIN
- EXPLAIN MONITOR
- EXPLICIT QUALIFIER USE
- INSERT
- LOAD
- LOCK TABLE
- MERGECOPY UTILITY
- MLS READWRITE
- MODIFY UTILITY
- MONITOR1
- MONITOR2
- OTHER
- PACKADM
- QUERY TUNING
- QUIESCE UTILITY
- READ
- RECOVER (UTILITY)
- RECOVER BSDS
- RECOVER INDOUBT
- REFERENCES
- RENAME INDEX
- RENAME TABLE
- REORG
- REPAIR
- REPAIR DBD UTILITY
- REPORT UTILITY
- RUNSTATS UTILITY
- SECADM
- SELECT
- SET ARCHIVE
- SQLADM
- START
- START DATABASE
- START DB2, STOP DB2, START DB(*) OR STOP DB(*)
- START PROFILE
- START RLIMIT
- STOP
- STOP DATABASE
- STOP OR START TRACE
- STOP PROFILE
- STOP RLIMIT
- STOPSPACE UTILITY
- SUBPKAUT
- SYSADM
- SYSCTRL
- SYSOPR
- SYSOPR SYSCTRL SYSADM SECADM
- TERM UTILITY
- TERMINATE UTILITY ON DATABASE
- TRIGGER
- UPDATE
- USAGE
- USE
- VALIDATE SECLABEL
- WRITE

Field Name: QW0140PR

**OBJECT**
The object type. N/A is printed if there is no object type. OBJECT can have the following values:
- ACEE
- APPLICATION PLAN
- BUFFERPOOL
- COLLECTION
- DATABASE
- DISTINCT TYPE
- FUNCTION
- SESSION VARIABLE
- JAR
- PACKAGE
- PROCEDURE
- ROLE
- ROW
- SCHEMA
- SEQUENCE
- STORAGE GROUP
- TABLE OR VIEW
- TABLESPACE
- TRUSTED CONTEXT
- USER AUTH (System privileges, such as SYSADM or SYSOPR)
- N/A

Field Name: QW0140OB

**OPTIONS**
The options that are used in the host to check the SQL statement. The bits of this field are used as indicators. If all bits are 0, the statement is not an SQL statement. OPTIONS has the following values:
- Bit 1 Host language character string delimiter
  - 0 Apostrophe
  - 1 Quote
- Bit 2 Decimal point symbol
  - 0 Period
  - 1 Comma
• Bit 3 SQL character string delimiter
  – 0 Apostrophe
  – 1 Quote
• Bit 4 Mixed character string indicator
  – 0 No
  – 1 Yes
• Bit 5 Host language options indicator
  – 0 Do not use host language options
  – 1 Use host language options
• Bits 6 to 8 Host language indicator
  – 001 Assembler
  – 010 COBOL
  – 011 PL/I
  – 100 None - Dynamic SQL
  – 101 FORTRAN
  – 110 COBOL II
  – 111 Null - See bits 17 to 24 for the language
• Bits 9 to 16 Character set that is being used
  – 00000000 Alphanumeric
  – 00000100 Katakana
• Bits 17 to 24 Alternate host language field
  – B Assembler
  – C COBOL
  – P PL/I
  – F Fortran 2
  – 2 COBOL II
  – 3 IBM COBOL
  – 4 C++
  – D C
• Bits 25 to 28 Time option
  – 0000 None
  – 1000 Local
  – 0100 JIS
  – 0010 ISO/EUR
  – 0001 USA
• Bits 29 to 32 Date option
  – 0000 None
  – 1000 Local
  – 0100 EUR
  – 0010 ISO/JIS
  – 0001 USA
• Bit 33 Decimal
  – 0 No
  – 1 Yes
• Bits 34 to 40 Unused
• Bits 41 to 48 Remote option
  – 00000001 SQL(ALL)
  – 00000010 SQL(Db2)
• Bits 49 to 56 SQL flag option
  – 00000000 No SQLFLAG option
  – 00000001 SQLFLAG(SAA)

Field Name: QW0140HO

SOURCE OBJECT
The source object name.
Field Name: QW0140SN

SOURCE OWNER
The source object owner.
Field Name: QW0140SC

TARGET OBJECT
The target object name.
Field Name: QW0140TC

TARGET OWNER
The target object owner.
Field Name: QW0140TN

SQL STMT
The SQL statement text. Long SQL text can be truncated.
Field Name: QW0140TX

ACEE UTOKEN
Shows the ACEE UTOKEN, if it is available. If it is not available, the first word of this field contains one of the following values:
  • UNABLE TO GET TOKEN
  • ABEND ACCESSING ACEE
Field Name: QW0140UT

RID OF ROW
Shows the row ID (RID) of the row that is updated or deleted if the table has multilevel security.
Field Name: QW0140UT

SECLABEL OF ROW
Shows the security label of the row for a table with multilevel security.
Field Name: QW0140UT

Example

AUTH Checked : USRT004
AuthID Type : PRIMARY OR SECONDARY AUTH ID
Reason : 0000000000 Statmnt Length : 000000176
Retcode : 00008 Priv Checked : SELECT
Object : TABLE OR VIEW Options : 'X'0400000000000000'
Source Object : SYSAUDITPOLICIES
Source owner : SYSIBM
Target Object:
  Target owner:
  SQL Stmt: SELECT AUDITPOLICYNAME, CHECKING, VALIDATE, OBJMAINT, EXECUTE, CONTEXT, SECMAINT, OBJECTSCHEMA, OBJECTTYPE, OBJECTNAME FROM SYSAUDITPOLICIES ORDER BY AUDITPOLICYNAME
ACEE UTOKEN : &.....A.L3C234 SLVM3 JESZ SLVM3 SYS1 RDR1 USRT004 SYS1
RID OF ROW:
SECLABEL OF ROW:
IFCID141: Audit DDL Grant/Revoke

The IFCID141 DD contains the extracted IFCID 141 (Audit DDL Grant/Revoke) records.

**Fields**

The IFCID141 DD has the following fields:

**GRANTOR/REVOKER**
The authorization ID of the user who gave or revoked the access.

Field Name: QW0141AC and QW0141OR

**REASON**
The reason why access was granted or revoked.

Field Name: QW0141RE

**RETURN**
The SQL return code.

Field Name: QW0141CO

**AUTHID TYPE**
The authorization ID type. AUTHID TYPE can have the following values:
- PRIMARY OR SECONDARY AUTH ID
- ROLE

Field Name: QW0141OT

**OBJECT**
The object type. OBJECT can have the following values:
- ACEE
- APPLICATION PLAN
- BUFFERPOOL
- COLLECTION
- DATABASE
- DISTINCT TYPE
- FUNCTION
- SESSION VARIABLE
- JAR
- PACKAGE
- PROCEDURE
- SCHEMA
- SEQUENCE
- STORAGE GROUP
- ROLE
- ROW
- TABLE OR VIEW
- TABLESPACE
Field Name: QW0141OB

OPTIONS
The options that were used in the host to check the SQL statement. The bits of this field are used as indicators. If all bits are 0, the statement is not an SQL statement. See the OPTIONS field of “IFCID140: Audit Authorization Failures” on page 1075 for a full description of the bits of this field.

Field Name: QW0141HO

SQL STMT
The SQL statement text. Long SQL text can be truncated.

Field Name: QW0141TX

Example

| GRANTOR : SYSADM | REASON : N/A | RETURN : 00000000562 |
| AUTHID TYPE : PRIMARY OR SECONDARY AUTH ID |
| OBJECT : APPLICATION PLAN OPTIONS: X'04000000000000000000000000000000' |
| SQL STMT: GRANT BIND, EXECUTE ON PLAN DSNTEP2 TO PUBLIC |

IFCID142: Audit DDL Create/Alter/Drop
The IFCID142 DD contains the extracted IFCID 142 (Audit DDL Create/Alter/Drop) records, which contain the record of CREATE, ALTER, and DROP statements against tables that are audited or that have multilevel security defined at the row level.

Fields

The IFCID142 DD has the following fields:

AUDIT DDL
ALTER, CREATE, or DROP.
Field Name: QW0142AC

TABLE NAME
The table name that is being created, altered, or dropped.
Field Name: QW0142TN

TABLE OWNER
The table owner (same as table qualifier).
Field Name: QW0142OW

TABLE OWNER TYPE
The table owner type. TABLE OWNER TYPE can have the following values:
• PRIMARY OR SECONDARY AUTH ID
• ROLE
TABLE CREATOR
The table creator.

OPTIONS
The options used in the host to check the SQL statement. The bits of this field are used as indicators. If all bits are 0, the statement is not an SQL statement. See the OPTIONS field of “IFCID140: Audit Authorization Failures” on page 1075 for a full description of the bits of this field.

DATABASE
The database ID.

TABLE OBID
The object ID.

SECLABEL OF MLS TABLE
The security label of the Multilevel Security (MLS) table.

MULTILEVEL SECURITY
Shows the multilevel security (MLS) status as follows:
• Y (on CREATE or DROP of an MLS Table, or ALTER to add a SECLABEL column)
• N (on ALTER of an MLS table)
• NON-MLS
• N/A

ROW/COLUMN ACCESS CONTROL
Shows the row and column access control status. The field can have one of the following values:
• ACTIVATE ROW ACCESS CONTROL
• ACTIVATE COLUMN ACCESS CONTROL
• ACTIVATE ROW AND COLUMN ACCESS CONTROL
• NO ACCESS CONTROL
• N/A

SQL STMT
The SQL statement text. Long SQL text can be truncated.

Example

Example
DATABASE: 00269
TABLE OBID: 00003
SECLABEL OF MLS TABLE: L3C234
MULTILEVEL SECURITY : N (on ALTER of an MLS table)
ROW/COLUMN ACCESS CONTROL: NO ACCESS CONTROL
SQL STMT: ALTER TABLE SCCCDB2A.TBCCDB2A ALTER COLUMN NAME SET DATA TYPE VARCHAR(10)

IFCID143: Audit First Write

The IFCID143 DD contains the extracted IFCID 143 (Audit First Write) records.

Fields

DATABASE
The database ID.
Field Name: QW0143DB

LOGRBA
The identifier of the unit of recovery.
Field Name: QW0143UR

PAGESET
The page set name or decimal identifier.
Field Name: QW0143PS

TABLE OBID
The object ID.
Field Name: QW0143OB

STATEMENT ID
The statement ID. This field can contain one of the following values:
• SQL statement ID
• 0
Field Name: QW0143SI

Example
DATABASE: 00269
LOGRBA : X'00000000000000036343'
PAGE SET: 00002
TABLE OBID: 00004
STATEMENT ID: 000000000000036343

IFCID144: Audit First Read

The IFCID144 DD contains the extracted IFCID 144 (Audit First Read) records.
The IFCID144 DD has the following fields:

**DATABASE**
The database ID.
Field Name: QW0144DB

**LOGRBA**
The identifier of the unit of recovery.
Field Name: QW0144UR

**PAGESET**
The page set name or decimal identifier.
Field Name: QW0144PS

**TABLE OBID**
The object ID.
Field Name: QW0144OB

**STATEMENT ID**
The statement ID. This field can contain one of the following values:
- SQL statement ID
- 0
Field Name: QW0144SI

Example

DATABASE: 00269  LOGRBA : X'00000000000001EE006FD'
PAGE SET: 00002  TABLE OBID: 00004
STATEMENT ID: 00000000000000000000

IFCID145: Audit DML Statement
The IFCID145 DD contains the extracted IFCID 145 (Audit DML Statement) records.

**Fields**

The IFCID145 DD has the following fields:

**LOCATION NAME**
The location name.
Field Name: QW0145LN

**PKG COLLECT ID**
The package collection identifier.
Field Name: QW0145PC
PROGRAM NAME
The program name.
Field Name: QW0145PN

TIME
The hexadecimal value of the precompiler timestamp.
Field Name: QW0145TS

TYPE
The SQL statement type.
Field Name: QW0145ST

STMT#
The precompiler statement number.
Field Name: QW0145SN

STATEMENT ID
The statement ID. This field can contain one of the following values:

- SQL statement ID
- 0

Field Name: QW0145SI

HOST OPTIONS
The options used in the host to check the SQL statement. The bits of this field are used as indicators. If all bits are 0, the statement is not an SQL statement. See the OPTIONS field of “IFCID140: Audit Authorization Failures” on page 1075 for a full description of the bits of this field.
Field Name: QW0145HO

SQL TEXT
The SQL statement text.
Field Name: QW0145RT_Var

DATABASE
The database ID.
Field Name: QW0145DB

TABLE OBID
The object ID.
Field Name: QW0145OB

ISOLATION
The isolation level of the DML statement. ISOLATION can have the following values:

- RR Repeatable read
- CS Cursor stability
- RS Read stability
- UR Uncommitted read
- RRX Repeatable read with X lock
- RSX Read stability with X lock

Field Name: QW0145IS

ACCESS SCHEMA
Field Name: QW0145AS_D
ACCESS OBJECT

The access control object name

Field Name: QW0145AO_D

Example

LOCATION NAME: STLEC1
PKG COLLCT ID: PKCC1A3
PROGRAM NAME : MCC1A3
TIME : 02:22:12
TYPE : INSERT
STMT#: 0000000014
STATEMENT ID : 000000000000036343
HOST OPTIONS : 0300D70010020000
DATABASE : 00269
TABLE OBID: 00004
SQL TEXT: INSERT INTO SCCC1A31 . TBCC1A31 ( ID , NAME ) VALUES ( 1 , 'X' )

IFCID269: Trusted Connection

The IFCID269 DD contains the extracted IFCID 269 (Trusted Connection) records.

Fields

The IFCID269 DD has the following fields:

CONNECTION TYPE

The type of trusted connection. CONNECTION TYPE can have the following values:

- ESTABLISHED
- REUSED

Field Name: QW0269TY

STATUS

The status of the trusted connection. This value indicates a success if a trusted connection was established or reused successfully. It indicates a failure if a trusted connection failed when it was tried to be established or reused. STATUS can have the following values:

- FAILED
- SUCCESS

Field Name: QW0269ST

SQLCODE

The SQLCODE that is returned after running the SQL statement.

Field Name: QW0269SQ

TRUSTED CONTEXT NAME

The trusted context name.

Field Name: QW0269TC

SYSTEM AUTHID USED

The system authorization ID that is used to establish the trusted connection.

Field Name: QW0269SA
ROLE ASSOCIATED
The default role that is associated with the context.
Field Name: QW0269RC

OBJECT OWNER
The owner of the objects that are created using the trusted context. This value indicates whether a ROLE or an AUTHORIZATION ID created the objects. OBJECT OWNER can have the following values:
- AUTHID
- ROLE
Field Name: QW0269OT

SECURITY LABEL
The security label.
Field Name: QW0269SL

TCP/IP ADDRESS
The communication TCP/IP address that is used for connection.
Field Name: QW0269AD

SERVAUTH NAME
The SERVAUTH name of the TCP/IP security zone.
Field Name: QW0269SR

ENCRYPTION
The encryption value that is to be associated with the encryption trust attribute for a trusted context.
Field Name: QW0269EC

JOB NAME
The job name for a local application.
Field Name: QW0269JN

REUSE AUTHID
The authorization ID under which a trusted connection is reused.
Field Name: QW0269RA

USER ROLE
The user role.
Field Name: QW0269RU

PROFILE NAME
The profile name.
Field Name: QW0269PR

Example

<table>
<thead>
<tr>
<th>CONNECTION TYPE</th>
<th>ESTABLISHED</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATUS</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>TRUSTED CONTEXT NAME</td>
<td>CTXCC1A33</td>
</tr>
<tr>
<td>SYSTEM AUTHID USED</td>
<td>USRT003</td>
</tr>
<tr>
<td>ROLE ASSOCIATED</td>
<td>ROLECC1A3_SYSADM</td>
</tr>
<tr>
<td>OBJECT OWNER</td>
<td>ROLE</td>
</tr>
<tr>
<td>SECURITY LABEL</td>
<td>........</td>
</tr>
<tr>
<td>TCP/IP ADDRESS</td>
<td>9.30.89.162</td>
</tr>
<tr>
<td>SERVAUTH NAME</td>
<td></td>
</tr>
<tr>
<td>ENCRYPTION</td>
<td>NONE</td>
</tr>
</tbody>
</table>
IFCID270: Trusted Context
The IFCID270 DD contains the extracted IFCID 270 (Trusted Context) records.

Fields

The IFCID270 DD has the following fields:

STATEMENT TYPE
The type of trusted context. STATEMENT TYPE can have the following values:
• ALTER
• CREATE

Field Name: QW0270TY

SQLCODE
The SQL return code from the CREATE or ALTER TRUSTED CONTEXT statement.

Field Name: QW0270SQ

SQL STMT LENGTH
The length of the SQL statement.

Field Name: QW0270SL

SQL STATEMENT
The SQL statement.

Field Name: QW0270SS

Example

| STATEMENT TYPE : CREATE   |
| SQLCODE : 0000000000     |
| SQL STMT LENGTH : 0000000236 |
| SQL STATEMENT : CREATE TRUSTED CONTEXT CTXCCIA33 BASED UPON CONNECTION USING SYSTEM AUTHID USRT003 ATTRIBUTES (ADDRESS '9.30.89 .162') DEFAULT ROLE ROLECC1A3_SYSADM WITH ROLE AS OBJECT OWNER AND QUALIFIER ENABLE WITH USE FOR USRT003 ROLE ROLECC1A3_SYSADM |

IFCID350: SQL Statement
The IFCID350 DD contains the extracted IFCID 350 (SQL Statement) records.

Fields

The IFCID350 DD has the following fields:
OPTIONS
Shows the parser options and host language.
Field Name: QW0350OT

HOST LANG
Shows the host language. HOST LANG can have the following values:
- ASSEMBLER
- COBOL
- C
- FORTRAN
- PL/I
- COBOL II
- IBM COBOL
- C++
- N/A
Field Name: QW0350HL

SOURCE CCSID
Field Name: QW0350CC

SQL SEGMENT
Shows a segment of the SQL statement. SQL SEGMENT can have the following values:
- FIRST The first segment of the SQL statement.
- LAST The last segment of the SQL statement.
- COMPLETE The entire SQL statement.
- OTHER Some other segment of the SQL statement.
Field Name: QW0350FG

SQL LENGTH
The total length of the SQL statement.
Field Name: QW0350TL

SQL STATEMENT
Shows the complete SQL statement that is being parsed or only a part of it.
Note: Host variables in this field are represented by :h..
Field Name: QW0350SP

SQL STATEMENT ID
Field Name: QW0350SI

SQL STATEMENT TYPE
Field Name: QW0350TY

Example
OPTIONS : X'04'    HOST LANG : N/A
SQL STATEMENT TYPE: DYNAMIC
SQL STATEMENT ID : 00000000000000000000
SOURCE CCSID : 00037
SQL SEGMENT : FIRST SQL LENGTH: 0000117158
SQL STATEMENT: INSERT INTO CCC501TAB ...
IFCID361: Audit Administrative Authorities

The IFCID361 DD contains the extracted IFCID 361 (Audit Administrative Authorities) records.

Fields

The IFCID361 DD has the following fields:

**AUTHORITY TYPE**

The type of administrative authority.

**ROLE or AUTHID**

The label that indicates the type of authorization followed by the role or authid that has the authority

Field Name: QW0361IT and QW0361ID_Var

**PRIVILEGE CHECKED**

The privilege that is checked. PRIVILEGE CHECKED can have the following values:

- ALL ON PACKAGES
- ALLPKAUT
- ALTER
- ALTER BUFFERPOOL
- ALTER INDEX
- ALTERTIN
- ARCHIVE
- BIND ADD
- BIND, REBIND OR FREE
- BINDAGENT
- CHECK DATA UTILITY
- CHECK UTILITY
- COMMENT ON
- COMMENT ON INDEX
- COPY
- COPY PACKAGE
- CREATE ALIAS
- CREATE GLOBAL TEMPORARY TABLE (CREATETMTAB)
- CREATE IN
- CREATE INDEX
- CREATE SECURE OBJECT
- CREATE STOGROUP
- CREATE SYNONYM
- CREATE TABLE
- CREATE TABLESPACE
• CREATE VIEW
• CREATEDBA
• CREATEDBC
• CREATEIN
• DBADM
• DBCtrl
• DBMAINT
• DDF COMMAND - START, STOP, OR CANCEL
• DEBUG SESSION
• DELETE
• DESCRIBE TABLE
• DIAGNOSE UTILITY
• DISPLAY
• DISPLAY ARCHIVE
• DISPLAY BUFFERPOOL
• DISPLAY DATABASE
• DISPLAY PROFILE
• DISPLAY RLIMIT
• DISPLAY THREAD OR DISPLAY DB
• DISPLAY UTILITY
• DROP
• DROP ALIAS
• DROP INDEX
• DROP SYNONYM
• DROPIN
• EXECUTE
• EXPLAIN
• EXPLAIN MONITOR
• EXPLICIT QUALIFIER USE
• INSERT
• LOAD
• LOCK TABLE
• MERGECOPY UTILITY
• MLS READWRITE
• MODIFY UTILITY
• MONITOR1
• MONITOR2
• OTHER
• PACKADM
• QUERY TUNING
• QUIESCE UTILITY
• READ
• RECOVER (UTILITY)
• RECOVER BSDS
• RECOVER INDOUBT
REFERENCES
  RENAME INDEX
  RENAME TABLE
  REORG
  REPAIR
  REPAIR DBD UTILITY
  REPORT UTILITY
  RUNSTATS UTILITY
  SECADM
  SELECT
  SET ARCHIVE
  SQLADM
  START
  START DATABASE
  START DB2, STOP DB2, START DB(*) OR STOP DB(*)
  START PROFILE
  START RLIMIT
  STOP
  STOP DATABASE
  STOP OR START TRACE
  STOP PROFILE
  STOP RLIMIT
  STOSPACE UTILITY
  SUBPKAUT
  SYSADM
  SYSCTRL
  SYSOPR
  SYSOPR SYSCTRL SYSADM SECADM
  TERM UTILITY
  TERMINATE UTILITY ON DATABASE
  TRIGGER
  UPDATE
  USAGE
  USE
  VALIDATE SECLABEL
  WRITE

Field Name: QW0361PR

SOURCE QUALIFIER
  The source qualifier.

Field Name: QW0361SC_Var

SOURCE OBJECT
  The source object name or original subsystem parameter value.

Field Name: QW0361SN_Var
TARGET QUALIFIER
The target qualifier.
Field Name: QW0361TC_Var

TARGET OBJECT
The target object name or modified z_parm value.
Field Name: QW0361TN_Var

OBJECT TYPE
The object type. OBJECT TYPE can have the following values:
- ACEE
- APPLICATION PLAN
- BUFFERPOOL
- COLLECTION
- DATABASE
- DISTINCT TYPE
- FUNCTION
- SESSION VARIABLE
- JAR
- PACKAGE
- PROCEDURE
- ROLE
- ROW
- SCHEMA
- SEQUENCE
- STORAGE GROUP
- TABLE OR VIEW
- TABLESPACE
- TRUSTED CONTEXT
- USER AUTH (System privileges, such as SYSADM or SYSOPR)
- N/A
Field Name: QW0361OT

OTHER OBJECT
The other object name or subsystem parameter value.
Field Name: QW0361ON_Var

SQL LENGTH
The length of the SQL statement.
Field Name: QW0361LL

SQL STATEMENT
The SQL statement (truncated at 4000 max).
Field Name: QW0361SQ
Example

AUTHORITY TYPE : U
AUTHID : USER000
PRIVILEGE CHECKED: STOP OR START TRACE
OBJECT TYPE : USER AUTH
SQL LENGTH : 0000000022
SQL STATEMENT : -STOP TRACE(AUDIT)

IFCID362: Trace Start and Stop with AUDITPOLICY

The IFCID362 DD contains the extracted IFCID 362 (Trace Start and Stop with AUDITPOLICY) records.

Fields

The IFCID362 DD has the following fields:

STATUS
The status of the trusted connection. This indicates a success if a trusted connection was established or reused successfully. It indicates a failure if a trusted connection failed when it was tried to be established or reused.

Field Name: QW0362ST

TYPE
The type

Field Name: QW0362TY

REASON CODE
The reason code.

Field Name: QW0362RN

CHECKING CATEGORY
The category CHECKING.

Field Name: QW0362CH

VALIDATE CATEGORY
The category VALIDATE.

Field Name: QW0362VA

OBJMAINT CATEGORY
The category OBJMAINT.

Field Name: QW0362OB

EXECUTE CATEGORY
The category EXECUTE.

Field Name: QW0362EX

CONTEXT CATEGORY
The category CONTEXT.

Field Name: QW0362CX

SECMAINT CATEGORY
The category SECMAINT.

Field Name: QW0362SM
Db2 START
Db2 startup.
Field Name: QW0362DS

DATABASE NAME
The Database name.
Field Name: QW0362DB

OBJECT TYPE
The object type.
Field Name: QW0362OT

AUDIT POLICY NAME
The audit policy name.
Field Name: QW0362AP_Var

TABLE SCHEMA NAME
The table schema name.
Field Name: QW0362TS_Var

TABLE NAME
The table name.
Field Name: QW0362TB_Var

SYSADMIN CATEGORY
The SYSADM categories.
Field Name: QW0362SA_Var

DBADMIN CATEGORY
The DBADM categories.
Field Name: QW0362DA_Var

COLLECTION ID
The collection ID.
Field Name: QW0362CO_Var

LIKE TABLES
The total number of tables that match the LIKE clause.
Field Name: QW0362TT

LIKE TABLES TRACED
The number of LIKE tables that are traced.
Field Name: QW0362TR

LIKE TABLE #
The name of the LIKE table that is traced.
Field Name: QW0362TN

Example

<table>
<thead>
<tr>
<th>STATUS</th>
<th>TYPE</th>
<th>REASON CODE</th>
<th>CHECKING CATEGORY</th>
<th>VALIDATE CATEGORY</th>
<th>OBJMAINT CATEGORY</th>
<th>EXECUTE CATEGORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUCCESS</td>
<td>START TRACE</td>
<td>00000000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SYSPRINT: Runtime messages and end-of-job summary

The SYSPRINT DD contains any warning and error messages that were generated during processing, plus a breakdown of the records that were read and processed.

Example

<table>
<thead>
<tr>
<th>DSN1SMFP - End of job summary</th>
<th>Read</th>
<th>Written</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total records:................</td>
<td>33</td>
<td>31</td>
</tr>
<tr>
<td>SMF Type 101 records:........</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Type 101 from DB2 Version 11:</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>IFCID 003 for distributed data:</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Other IFCID 003:...........</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Other Type 101 IFCIDs:.......</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Type 101 from other DB2 releases:</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SMF Type 102 records:........</td>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td>Type 102 from DB2 Version 11:</td>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td>IFCID 004:...................</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>IFCID 005:...................</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>IFCID 023:...................</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>IFCID 024:...................</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>IFCID 025:...................</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>IFCID 083:...................</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>IFCID 106:...................</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>IFCID 140:...................</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>IFCID 141:...................</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>IFCID 142:...................</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>IFCID 143:...................</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>IFCID 144:...................</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>IFCID 145:...................</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>IFCID 269:...................</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>IFCID 270:...................</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>IFCID 350:...................</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>IFCID 361:...................</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>IFCID 362:...................</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Other Type 102 IFCIDs:.......</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Type 102 from other DB2 releases:</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Other SMF records:...............</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Invalid IFCID Records:........</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Attention: This figure indicates the number of input records that were counted under a particular IFCID type but that were subsequently rejected because subsequent processing has revealed problems such as missing sections. The Invalid IFCID Records count, therefore, is not part of the Total Records Read count.
Appendix D. DSNADMSB

The DSNADMSB program collects information about a Db2 subsystem and its objects and applications. It calls the ADMIN_INFO_SQL stored procedure.

Tip: You can run the DSNTEJ6I sample job that is supplied with Db2 in the prefix.SDSNSAMP data set to run this program. For more information, see DSNTEJ6I (Db2 Programming samples).

IBM Support can use the information that DSNADMSB gathers to duplicate a customer environment to diagnose and resolve problems. This capability is especially useful for re-creating performance problems.

Output

Output from the DSNADMSB program consists of files that contain one or more of the following items:

- Data definition statements for re-creating user objects
- Statistics from the Db2 catalog
- INSERT statements for inserting rows into tables:
  - DSN_PROFILE_ATTRIBUTES
  - DSN_PROFILE_TABLE
  - PLAN_TABLE
  - SYSACCELERATEDTABLES
  - SYSACCELERATORS
- Output from:
  - DSN_DETCOST_TABLE
  - DSN_PREDICAT_TABLE
  - DSN_PROFILE_TABLE
  - DSN_PROFILE_ATTRIBUTES
  - PLAN_TABLE
  - SYSACCELERATEDTABLES
  - SYSACCELERATORS
- Subsystem parameter settings and module entry point list (MEPL) information
- Module entry point list (MEPL) information that relates to query processing
- Status information from stored procedure ADMIN_INFO_SQL

The output from ADMIN_INFO_SQL is intended primarily for the use of IBM Support. The format and content of the output might change at any time.

Authorization required

To execute DSNADMSB, you must have the following Db2 privileges or authorities:

- EXECUTE authority on plan DSNADMSB
- One of the following privileges or authorities:
  - The EXECUTE privilege on the ADMIN_INFO_SQL stored procedure
  - Ownership of the ADMIN_INFO_SQL stored procedure
  - SYSADM authority

In addition, if you direct DSNADMSB to write its output to data sets, you need to be authorized to perform either of the following tasks:

- Create data sets
• Write to already existing data sets

Related tasks:

- Collecting service SQL documentation

Related reference:

- ADMIN_INFO_SQL stored procedure (Db2 SQL)
- DSNTEJ6I (Db2 Programming samples)

---

**Parameters of the DSNADMSB program**

The parameters of the DSNADMSB program determine the types of information that the program gathers, and the destination to which the information is written.

The output from DSNADMSB is primarily for the use of IBM Software Support. The output might change at any time.

All parameters are positional and required, and no parameter value can be NULL. Specify parameters in a data set that is associated with the INPUTP DD statement, with one parameter value in each record. A single parameter can span multiple records. Indicate continuation of the text for a parameter with a plus sign (+) in column one of all records after the first one. DSNADMSB trims blanks at the end of all lines. DSNADMSB does not trim blanks within the parameter text.

**Parameter descriptions**

**table-creator**

The qualifier for the table, table list table (LIST_TABLE-list-table), or PLAN_TABLE for which DSNADMSB gathers information. The maximum length of table-creator is 128 bytes. table-creator cannot be null.

**table-name**

One of the following values:

- The name of a single user object that has table-creator as its qualifier. The name must identify one of the following types of objects:
  - Base table
  - View
  - Alias
  - Clone table
  - Created temporary table
  - History table
  - Materialized query table
  - Implicitly created table for an XML column
- LIST_TABLE-list-table
  list-table identifies a Db2 table on the local subsystem that has table-creator as the qualifier. The table must contain two columns, named CREATOR and TABLE. Each row of list-table identifies a table or view for which DSNADMSB is to gather information. The types of tables that are specified in LIST_TABLE-list-table are the same as the types of tables that can be specified by table-name.
- PLAN_TABLE
Specifies that DSNADMSB uses table-creator.PLAN_TABLE and the plan-info parameter to determine the tables about which information is gathered. The referenced PLAN_TABLE must be a base table.

**Recommendation:** When you specify PLAN_TABLE, ensure that the view reference table, DSN_VIEWREF_TABLE, exists before you execute EXPLAIN. The existence of DSN_VIEWREF_TABLE is especially important if the queries about which you are collecting information contain views. If DSN_VIEWREF_TABLE exists and has been populated by EXPLAIN, you can use DSNADMSB to collect view data that is specific to the queries of interest, instead of collecting data on all view dependencies. The DSN_VIEWREF_TABLE must have the same qualifier as the plan table. This qualifier is the first DSNADMSB input parameter, table-creator.

The maximum length of **table-name** is 128 bytes.

If the table name is a delimited identifier, do not include the delimiters in **table-name**.

**catalog-creator**

The qualifier for the Db2 catalog, or DEFAULT, which indicates that the catalog qualifier is SYSIBM.

The maximum length of **catalog-creator** is 128 bytes.

**plan-info**

If the table that is specified by **table-name** is PLAN_TABLE, **plan-info** is a value of the following form, or NONE:

```
program-begin-queryno-end-queryno
```

The meanings of these variables are:

**program**

A value from the PROGNAME column of the PLAN_TABLE, or a pattern that specifies a set of PROGNAME values in the PLAN_TABLE. Any pattern that is valid in a LIKE predicate can be specified. **program** represents programs or packages for which DSNADMSB collects PLAN_TABLE information.

**begin-queryno**

A value from the QUERYNO column of the PLAN_TABLE. The value represents the lowest statement number for which DSNADMSB collects PLAN_TABLE information.

**end-queryno**

A value from the QUERYNO column of the PLAN_TABLE. The value represents the highest statement number for which DSNADMSB collects PLAN_TABLE information.

If **table-name** does not specify PLAN_TABLE, the **plan-info** value must be NONE.

The maximum length of **plan-info** is 150 bytes.

**collect-ddl**

Specifies whether DSNADMSB returns the data definition language statements that were used to create the input tables or views that are specified by **table-name**, and data definition language statements for related objects. The length of **collect-ddl** is 1 byte.

When the input table is not PLAN_TABLE, possible values are:
N  Do not return the data definition language statements that created the objects.

Y  Return the data definition language statements that created:
   • The input objects
   • Foreign keys that reference the input objects
   • Views on the input objects

0  Return the data definition language statements that created:
   • The input objects. Statements that create views on the input objects or foreign keys that reference the input objects are not collected.

1  Return the data definition language statements that created:
   • The input objects
   • Views on the input objects

2  Return the data definition language statements that created:
   • The input objects
   • Foreign keys that reference the input objects

3  Return the data definition language statements that created:
   • The input objects
   • Foreign keys that reference the input objects
   • Views on the input objects
   • Other objects that depend on the input objects, such as materialized query tables

   This option can result in a large amount of data. Do not use this option for data collection that is requested by IBM Support.

4  Return the same data definition language statements that are returned when option Y is specified.

When the input table is PLAN_TABLE, possible values are:

N  Do not return the data definition language statements that created the objects.

Y  Return the data definition language statements that created:
   • The objects that are identified by plan-info
   • Foreign keys that reference the objects that are identified by plan-info
   • If DSN_VIEWREF_TABLE exists and is populated, views or materialized query tables that are used to process the queries that are identified by plan-info.
   • If DSN_VIEWREF_TABLE does not exist, views on objects that are identified by plan-info. DSNAADMSB requires more time to gather data if DSN_VIEWREF_TABLE is not available than if DSN_VIEWREF_TABLE is available.

0  Return the data definition language statements that created:
   • The objects that are identified by plan-info only. Statements that create views on the objects or foreign keys that reference the objects that are identified by plan-info are not collected.

1  Return the data definition language statements that created:
   • The objects that are identified by plan-info
• If DSN_VIEWREF_TABLE exists and is populated, views or materialized query tables that are used to process the queries that are identified by plan-info.

• If DSN_VIEWREF_TABLE does not exist, views on objects that are identified by plan-info. DSNADMSB requires more time to gather data if DSN_VIEWREF_TABLE is not available than if DSN_VIEWREF_TABLE is available.

2 Return the data definition language statements that created:
• Foreign keys that reference the objects that are identified by plan-info

3 Return the data definition language statements that created:
• The objects that are identified by plan-info
• Foreign keys that reference the objects that are identified by plan-info
• Views on objects that are identified by plan-info
• Other objects that depend on the objects that are identified by plan-info, such as materialized query tables

This option can result in a large amount of data. Do no specify this option for problem analysis by IBM Software Support unless they direct you to do so.

4 Return the data definition language statements that created:
• The objects that are identified by plan-info
• Foreign keys that reference the objects that are identified by plan-info
• Views on objects that are identified by plan-info

This option does not use information from DSN_VIEWREF_TABLE. DSNADMSB requires more time to gather data if you choose option 4 than if you choose option Y, and DSN_VIEWREF_TABLE is available.

collect-stats
Specifies whether DSNADMSB returns statistical information from Db2 catalog tables about the tables that are specified by table-name and related objects. Possible values are:

Y Return statistical information about tables from the Db2 catalog.

N Do not return statistical information about tables from the Db2 catalog.

Important: Setting a collect-stats value of Y might generate large amounts of data. Set collect-stats to N unless you specifically need statistical information from Db2 catalog tables.

The length of collect-stats is 1 byte.

collect-colstats
Specifies whether DSNADMSB returns statistical information from Db2 catalog tables about the columns in tables that are specified by table-name and related objects. Possible values are:

Y Return statistical information about columns from the Db2 catalog.

N Do not return statistical information about columns from the Db2 catalog.

Important: Setting a collect-colstats value of Y might generate large amounts of data. Set collect-colstats to N unless you specifically need statistical information from Db2 catalog tables.
The length of `collect-colstats` is 1 byte.

**edit-dll**
Specifies whether DSNADMSB modifies the data definition language statements that it generates so that the data definition language statements can be more easily executed by IBM Support. Examples of changes that DSNADMSB makes are:
- Setting the STOGROUP to SYSDEFLT
- Setting PRIQTY and SECQTY to their minimum values
- Setting DEFINE to NO
- Commenting out foreign key definitions
Possible values are:

**Y** Edit the data definition language statements that DSNADMSB produces.

Y is the recommended value if you do not send data to populate the tables that are specified by `table-name` to IBM Support.

**N** Do not edit the data definition language statements that DSNADMSB produces.

The length of `edit-dll` is 1 byte.

**edit-version-mode**
Specifies that the output should be formatted for a different version and mode of Db2 for z/OS than the version and mode that is currently running when collecting information. You must specify the version number and the mode, or you can specify NONE so that the output is not converted to another format.

Valid values for mode are `C` for conversion mode and `N` for new-function mode. For example, if your Db2 subsystem is running in DB2 10 conversion mode, and you want to generate the output for DB2 9 new-function mode, specify `9-N`.

This is an input parameter of type CHAR(4) and cannot be null.

**partition-rotation**
Specifies whether DSNADMSB checks the amount of rotation that a partitioned table has undergone. DSNADMSB determines the number of partition rotations that are needed to synchronize logical partitions with physical partitions.
Possible values are:

**Y** Check for the amount of partition rotation.

Y is valid only for partitioned tables.

**N** Do not check for the amount of partition rotation.

The length of `partition-rotation` is 1 byte.

**output-method**
Specifies the type of destination for DSNADMSB output. Possible values are:

**R** Output is returned in the job stream. In most cases, R should be used.

**Q** Output is returned in data sets that DSNADMSB creates. You supply the qualifier name and primary and secondary allocation quantities for those data sets in the `output-info` parameter.
Important: The data sets are temporary data sets that are created on scratch packs. Depending on how the z/OS system is configured, the data sets might be deleted after a short time.

N Output is returned in existing data sets that are allocated by the WLM environment startup procedure. You supply the data set names in the output-info parameter.

D Output is returned in data sets that DSNADMSB creates. You supply the data set names and primary and secondary allocation quantities for those data sets in the output-info parameter.

The length of output-method is 1 byte.

output-info
Specifies output data set information. The information depends on the value of output-method:

<table>
<thead>
<tr>
<th>output-method value</th>
<th>output-info value</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>NONE</td>
</tr>
<tr>
<td>Q</td>
<td>A string of this form:</td>
</tr>
<tr>
<td></td>
<td>qualifier-primary-secondary</td>
</tr>
<tr>
<td></td>
<td>A string of up to 29 bytes, or DEFAULT. DSNADMSB appends a string that defines the type of output data set. If qualifier is not DEFAULT, qualifier must conform to the rules for z/OS data set names. If qualifier is DEFAULT, DSNADMSB generates a qualifier value of the following form:</td>
</tr>
<tr>
<td></td>
<td>PMnnnnn.yymmdd.Thhmmss</td>
</tr>
<tr>
<td></td>
<td>nnnnn is the PMR number. yymmdd and hhmmss are the date and time when DSNADMSB ran.</td>
</tr>
<tr>
<td></td>
<td>The strings that DSNADMSB appends to qualifier are:</td>
</tr>
<tr>
<td></td>
<td>1 .DDL for the data set that contains data definition statements for user tables or the PLAN_TABLE</td>
</tr>
<tr>
<td></td>
<td>1 .SQL for the data set that contains SQL statements that populate PLAN_TABLE, DSN_PROFILE_TABLE, DSN_PROFILE_ATTRIBUTES, and SYSACCELERATORS</td>
</tr>
<tr>
<td></td>
<td>1 .STATS for the data set that contains SQL statements that populate Db2 catalog tables with statistical information about tables</td>
</tr>
<tr>
<td></td>
<td>1 .COLST for the data set that contains SQL statements that populate Db2 catalog tables with statistical information about columns</td>
</tr>
<tr>
<td></td>
<td>1 .EXPL for the data set that contains output from tables PLAN_TABLE, DSN_PREDICAT_TABLE, DSN_DETDCOST_TABLE, DSN_PROFILE_TABLE, DSN_PROFILE_ATTRIBUTES, and SYSACCELERATORS</td>
</tr>
<tr>
<td></td>
<td>1 .PARM for the data set that contains subsystem parameter settings</td>
</tr>
<tr>
<td></td>
<td>primary The primary allocation quantity for the output data sets, or DEFLT. If you specify DEFLT, the primary allocation quantity is 200 tracks.</td>
</tr>
<tr>
<td></td>
<td>secondary The secondary allocation quantity for the output data sets, or DEFLT. If you specify DEFLT, the secondary allocation quantity is 200 tracks.</td>
</tr>
<tr>
<td>output-method value</td>
<td>output-info value</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>N</td>
<td>A string of this form:</td>
</tr>
<tr>
<td></td>
<td><code>ddldd-sqldd-statsdd-colstdd-colstdd-parmdd</code></td>
</tr>
</tbody>
</table>

Each part of the string is the DD name for a data set that is defined in the WLM startup procedure for the WLM environment in which the ADMIN_INFO_SQL stored procedure runs. The DD names are:

- `ddldd` for the data set that contains data definition statements for user tables or the `PLAN_TABLE`
- `sqldd` for the data set that contains SQL statements that populate `PLAN_TABLE`, `DSN_PROFILE_TABLE`, `DSN_PROFILE_ATTRIBUTES`, and `SYSACCELERATORS`
- `statsdd` for the data set that contains SQL statements that populate Db2 catalog tables with statistical information about tables
- `colstdd` for the data set that contains SQL statements that populate Db2 catalog tables with statistical information about columns
- `expldd` for the data set that contains output from tables `PLAN_TABLE`, `DSN_PREDICAT_TABLE`, `DSN_DETTCOST_TABLE`, `DSN_PROFILE_TABLE`, `DSN_PROFILE_ATTRIBUTES`, and `SYSACCELERATORS`
- `parmdd` for the data set that contains subsystem parameter settings
output-method value  output-info value

D

Records of this form:

DDL;dsname;volser;allocunits;primary;secondary-
  +SQL;dsname;volser;allocunits;primary;secondary-
  +STATS;dsname;volser;allocunits;primary;secondary-
  +COLST;dsname;volser;allocunits;primary;secondary-
  +EXPL;dsname;volser;allocunits;primary;secondary-
  +PARM;dsname;volser;allocunits;primary;secondary

The plus (+) signs are not part of the output-info value. You need to put a plus (+) sign in column 1 when you continue the output-info value in a new record.

The meanings of the items in the string are:

**DDL, SQL, STATS, COLST, EXPL, PARM**

Identifies the type of output that DSNADMSB puts in the data set:

- **DDL** for the data set that contains data definition statements for user tables or the PLAN_TABLE
- **SQL** for the data set that contains SQL statements that populate PLAN_TABLE, DSN_PROFILE_TABLE, DSN_PROFILE_ATTRIBUTES, and SYSACCELERATORS
- **STATS** for the data set that contains SQL statements that populate Db2 catalog tables with statistical information about tables
- **COLST** for the data set that contains SQL statements that populate Db2 catalog tables with statistical information about columns
- **EXPL** for the data set that contains output from tables PLAN_TABLE, DSN_PREDICAT_TABLE, DSN_DETCOST_TABLE, DSN_PROFILE_TABLE, DSN_PROFILE_ATTRIBUTES, and SYSACCELERATORS
- **PARM** for the data set that contains subsystem parameter settings

**dbname**

The fully qualified name of the data set that DSNADMSB allocates, or DEFAULT. If you specify DEFAULT, the data set names have this form:

PMnnnnn.Dyyymmd.Tthhmss.Vn.COLST
PMnnnnn.Dyyymmd.Tthhmss.Vn.DDL
PMnnnnn.Dyyymmd.Tthhmss.Vn.EXPL
PMnnnnn.Dyyymmd.Tthhmss.Vn.PARM
PMnnnnn.Dyyymmd.Tthhmss.Vn.SQL
PMnnnnn.Dyyymmd.Tthhmss.Vn.STATS

nnnnn is the PMR number, without the branch code or country code.

yymmdd is the date and hhmmss is the time when DSNADMSB ran.

n is a release indicator.

**volser**

The volume serial on which the data set is created.

**allocunits**

Valid values are TRK or CYL.

**primary**

The primary allocation quantity for the output data set.

**secondary**

The secondary allocation quantity for the output data set.

The maximum length of output-info is 1024 bytes.

**pmr-info**

The number of the PMR for which the data is being collected, in this form:

number.branch-code.country-code
Before running DSNADMSB

Certain activities might be required before you run the DSNADMSB program.

Before you run DSNADMSB, perform the following actions:

- Check that the SYSPROC.ADMIN_INFO_SQL stored procedure that is supplied by Db2 is installed. In general, this activity is performed during the installation process.
  Installation job DSNTIJRT installs all Db2-supplied routines and sets up the WLM environments for them.
- Check that the plan for DSNADMSB is bound. In general, this activity is performed during the installation process. Installation job DSNTIJRT binds the package and plan for DSNADMSB.
- Ensure that enough space is available for the output. DSNADMSB might generate large amounts of data. The average is 2 - 3 MB of space, but larger workloads might generate up to 20 MB of data.
- Prepare a job for running DSNADMSB. The easiest way to do that is to customize a copy of sample job DSNTEJ6J, which is in data set prefix.SDSNSAMP. The job prolog has detailed instructions on how to customize the job.

**Recommendation:** If you are running DSNADMSB to collect information from the plan table, `PLAN_TABLE`, ensure that the view reference table, `DSN_VIEWREF_TABLE`, also exists before you execute EXPLAIN. The existence of `DSN_VIEWREF_TABLE` is especially important if the queries about which you are collecting information contain views. If `DSN_VIEWREF_TABLE` exists and has been populated by EXPLAIN, you can use DSNADMSB to collect view data that is specific to the queries of interest, instead of collecting data on all view dependencies. The `DSN_VIEWREF_TABLE` must have the same qualifier as the plan table. This qualifier is the first DSNADMSB input parameter, `table-creator`.

**Related concepts:**
- Job DSNTIJRT (Db2 Installation and Migration)

**Related reference:**
- “Examples of DSNADMSB invocation” on page 1110
- “Data sets that DSNADMSB uses” on page 1109
- “Parameters of the DSNADMSB program” on page 1100
- Job DSNTIJSG (Db2 Installation and Migration)
- SYSPROC.ADMIN_INFO_SQL stored procedure (Db2 SQL)
- DSN_VIEWREF_TABLE (Db2 Performance)
Data sets that DSNADMSB uses

The DSNADMSB utility uses a number of data sets during its operation.

DSNADMSB runs under the DSN Db2 command processor. The following table lists the data sets that DSNADMSB uses, in addition to the standard data sets that are required for running an application under DSN.

Table 150. Data sets that DSNADMSB uses

<table>
<thead>
<tr>
<th>Data set type</th>
<th>DD name</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>INPUTP</td>
<td>Contains the DSNADMSB parameters. The INPUTP data set must have a logical record length of 80. Only bytes 1 through 71 can contain input data.</td>
<td>Yes</td>
</tr>
<tr>
<td>Output</td>
<td>None</td>
<td>Up to six data sets that contain the diagnostic data that DSNADMSB generates. The data set names and characteristics are determined by the values that you specify for the output-method and output-info input parameter values.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Copying the data that DSNADMSB and ADMIN_INFO_SQL collect to another subsystem

The jobs that DSNADMSB and ADMIN_INFO_SQL produce are primarily intended for the use of IBM Software Support. However, you can modify those jobs so that you can run them on your own test systems to reproduce a problem environment.

Procedure

1. Ensure that your test system does not contain data that conflicts with the data that DSNADMSB or ADMIN_INFO_SQL collects.

   Recommendation: Use a newly installed Db2 subsystem for testing.

2. Customize the jobs:
   a. Modify the JOB statement for your test system.
   b. Change the subsystem name to the subsystem name for your test system.
   c. Change the steps that run TSOBATCH so that they run IKJEFT01. For example, suppose that the original code looks like this:
      ```
      //SETUP   EXEC TSOBATCH,DB2LEV=DB2A
      ```
      You need to change the code to something like this:
      ```
      //SETUP   EXEC PGM=IKJEFT01,DYNAMNBR=20
      ```
   d. Change the steps that run DSNTEP3 so that they run DSNTEP2. For example, suppose that the original code looks like this:
You need to change the code to something like this:

//SYSTSIN DD *
DSN S(SSTR) R(1) T(1)
RUN Program(DSNTEP3)
END

3. Set the subsystem parameters on the test system as indicated in the output file that contains subsystem parameter values (PARAM file).

If DSNADMSB or ADMIN_INFO_SQL output is returned in a single job stream (output-method is R), the subsystem parameter output is at the end of the job output.

4. In the data definition language job (the job that contains **DDL Information **), make these changes:
   a. Modify the -ALTER BUFFERPOOL statements at the end of the job to contain settings that are appropriate for your test system.
   b. Remove the asterisks (**) in front of the -ALTER BUFFERPOOL statements.
   c. Move the -ALTER BUFFERPOOL statements to the place in the job that says:

      **BUFFERPOOL INFORMATION GOES HERE

5. Run the remainder of the jobs, in the following order:
   a. Data definition language job (contains **DDL Information **)
   b. Statistics INSERTs job (contains **Stats inserts **)
   c. Column statistics INSERTs job (contains **Column Stats inserts **)
   d. Plan, profile and accelerator INSERTs job (contains **Plan, Profile and Accelerator Inserts **)

---

**Examples of DSNADMSB invocation**

Use the DSNADMSB invocation examples as models for generating your own DSNADMSB output.

**Example: Collecting data from a PLAN_TABLE**

Suppose that you want DSNADMSB to retrieve data from plan table
SYSADM.PLAN_TABLE rows for which PROGNAME is APROGRAM and 1<=QUERYNO<=12345. You want DSNADMSB to create the output in data sets whose names and characteristics you specify. The parameter values that you specify are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>table-creator</td>
<td>SYSADM</td>
<td>These two parameters direct DSNADMSB to collect data from SYSADM.PLAN_TABLE.</td>
</tr>
<tr>
<td>table-name</td>
<td>PLAN_TABLE</td>
<td></td>
</tr>
<tr>
<td>catalog-creator</td>
<td>DEFAULT</td>
<td>For catalog queries, directs DSNADMSB to use the default catalog table qualifier of SYSIBM.</td>
</tr>
</tbody>
</table>
### Parameter | Value | Explanation
---|---|---
plan-info | APROGRAM-1-12345 | Directs DSNADMSB to collect data only for rows for which PROGNAME is 'APROGRAM' and QUERYNO is between 1 and 12345, inclusive.

collect-dnl | Y | Directs DSNADMSB to collect the data definition statements that created SYSADM.PLAN_TABLE and associated objects.

collect-stats | Y | Directs DSNADBMSB to collect statistics about tables from the Db2 catalog.

collect-colstats | N | Directs DSNADBMSB not to collect statistics about table columns from the Db2 catalog.

edit-dnl | Y | Directs DSNADMSB to modify the data definition language statements that it generates so that the data definition language statements can be more easily executed by IBM Support.

edit-version-mode | NONE | Directs DSNADMSB not to modify its output to run on a different Db2 version from the version for which the data was collected.

partition-rotation | N | Directs DSNADMSB not to check partition rotation.

output-method | D | These two parameters direct DSNADMSB to write data to output data sets that DSNADMSB allocates on volume ESDMP, with the specified data set names and space allocations.

output-info | DDL,SYSADM.DDL.P12345;EDSDMP;TRK;200;200-+SQL,SYSADM.SQL.P12345;EDSDMP;TRK;200;200-+STATS,SYSADM.STATS.P12345;EDSDMP;TRK;200;200-+COLST;DEFAULT;EDSDMP;TRK;50;50-+EXPL;SYSADM.EXPL.P12345;EDSDMP;TRK;200;200-+P ARM;SYSADM.PARM.P12345;EDSDMP;TRK;200;200 | This is the PMR number for the problem that requires data collection.

pmr-info | 12345.000.000 | 

The JCL for the step that executes DSNADMSB looks like this:
```java
//DSNADMSB EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//SYSTSPRT DD SYSOUT=*  
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=4096)  
//SYSUDUMP DD SYSOUT=*  
//SYSTSIN DD *  
   DSN SYSTEM(dsn)  
   RUN PROGRAM(DSNADMSB) PLAN(DSNADMSB)  
//INPUTP DD *  
   SYSADM  
   PLAN_TABLE  
   DEFAULT  
   APROGRAM-1-12345  
   Y  
   Y
```
Example: Collecting data for all rows in a PLAN_TABLE

Suppose that you want DSNADMSB to retrieve data about all rows in a PLAN_TABLE. You want DSNADMSB to generate the names for the output data sets. The generated names are:

- PM11111.Dyymmdd.Dhhmmss.COLST
- PM11111.Dyymmdd.Dhhmmss.DDL
- PM11111.Dyymmdd.Dhhmmss.EXPL
- PM11111.Dyymmdd.Dhhmmss.PARM
- PM11111.Dyymmdd.Dhhmmss.SQL
- PM11111.Dyymmdd.Dhhmmss.STATS

11111 is the PMR number. yymmdd and hhmss are the date and time when DSNADMSB ran.

The parameter values that you specify are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>table-creator</td>
<td>SYSADM</td>
<td>These two parameters direct DSNADMSB to collect data about SYSADM.PLAN_TABLE.</td>
</tr>
<tr>
<td>table-name</td>
<td>PLAN_TABLE</td>
<td></td>
</tr>
<tr>
<td>catalog-creator</td>
<td>DEFAULT</td>
<td>For catalog queries, directs DSNADMSB to use the default catalog table qualifier of SYSIBM.</td>
</tr>
<tr>
<td>plan-info</td>
<td>%-%0-999999</td>
<td>This parameter tells DSNADMSB to collect data for all rows in SYSADM.PLAN_TABLE by requesting all rows for which PROGNAME=’%’ and 0&lt;=QUERYNO&lt;=999999.</td>
</tr>
<tr>
<td>collect-dll</td>
<td>Y</td>
<td>Directs DSNADMSB to collect the data definition statements that created SYSADM.PLAN_TABLE and associated objects.</td>
</tr>
<tr>
<td>collect-stats</td>
<td>Y</td>
<td>Directs DSNADBMSB to collect statistics about tables from the Db2 catalog.</td>
</tr>
<tr>
<td>collect-colstats</td>
<td>N</td>
<td>Directs DSNADBMSB not to collect statistics about table columns from the Db2 catalog.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
<td>Explanation</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>edit-ddl</td>
<td>Y</td>
<td>Directs DSNADMSB to modify the data definition language statements that it generates so that the data definition language statements can be more easily executed by IBM Support.</td>
</tr>
<tr>
<td>edit-version-mode</td>
<td>NONE</td>
<td>Directs DSNADMSB not to modify its output to run on a different Db2 version from the version for which the data was collected.</td>
</tr>
<tr>
<td>partition-rotation</td>
<td>N</td>
<td>Directs DSNADMSB not to check partition rotation.</td>
</tr>
<tr>
<td>output-method</td>
<td>Q</td>
<td>Q directs DSNADMSB to write output data to data sets that DSNADMSB creates. All output data sets have a data set qualifier of SYSADM, a primary allocation quantity of 200, and a secondary allocation quantity of 200.</td>
</tr>
<tr>
<td>output-info</td>
<td>DEFAULT-200-200</td>
<td></td>
</tr>
<tr>
<td>pmr-info</td>
<td>11111.000.000</td>
<td>This is the PMR number for the problem that requires data collection.</td>
</tr>
</tbody>
</table>

The JCL for the step that executes DSNADMSB looks like this:

```
//DSNADMSB EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//SYSTSPRT DD SYSOUT=*  
//SYSPRINT DD SYSOUT**,DCB=(RECFM=VB,LRECL=4096)  
//SYSUDUMP DD SYSOUT**  
//SYSTSIN DD *  
  DSN SYSTEM(DSN)  
  RUN PROGRAM(DSNADMSB) PLAN(DSNADMSB)  
//INPUTP DD *  
  SYSADM  
  PLAN TABLE  
  DEFAULT  
  %-0-999999  
  Y  
  Y  
  N  
  Y  
  NONE  
  N  
  Q  
  DEFAULT-200-200  
  11111.000.000
```

Example: Collecting data using a table list as input

Suppose that you want DSNADMSB to retrieve data about the DSN8B10.EMP table and the DSN8B10.VDEPT view. You want DSNADMSB to send the output to the job stream.

Before you run DSNADMSB, you need to create a Db2 table with a CREATOR and a TABLE column, and insert rows that contain the qualifiers and names of the tables or views for which you want DSNADMSB to collect data. Use SQL statements like these:
CREATE TABLE TL1 (CREATOR VARCHAR(128),
                    TABLE VARCHAR(128));
INSERT INTO TL1 VALUES ('DSN8B10','EMP');
INSERT INTO TL1 VALUES ('DSN8B10','VDEPT');

The parameter values that you specify are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>table-creator</td>
<td>SYSADM</td>
<td>These two parameters direct DSNADMSB to collect data about the tables whose names are in table SYSADM.LIST_TABLE-TL1.</td>
</tr>
<tr>
<td>table-name</td>
<td>LIST_TABLE-TL1</td>
<td>For catalog queries, directs DSNADMSB to use the default catalog table qualifier of SYSIBM.</td>
</tr>
<tr>
<td>catalog-creator</td>
<td>DEFAULT</td>
<td>This value must be NONE, because PLAN_TABLE data is not being collected.</td>
</tr>
<tr>
<td>plan-info</td>
<td>NONE</td>
<td>This value must be NONE, because PLAN_TABLE data is not being collected.</td>
</tr>
<tr>
<td>collect-dll</td>
<td>Y</td>
<td>Directs DSNADMSB to collect the data definition statements that created SYSADM.PLAN_TABLE and associated objects.</td>
</tr>
<tr>
<td>collect-stats</td>
<td>Y</td>
<td>Directs DSNADBMSB to collect statistics about tables from the Db2 catalog.</td>
</tr>
<tr>
<td>collect-colstats</td>
<td>N</td>
<td>Directs DSNADBMSB not to collect statistics about table columns from the Db2 catalog.</td>
</tr>
<tr>
<td>edit-dll</td>
<td>Y</td>
<td>Directs DSNADMSB to modify the data definition language statements that it generates so that the data definition language statements can be more easily executed by IBM Support.</td>
</tr>
<tr>
<td>edit-version-mode</td>
<td>NONE</td>
<td>Directs DSNADMSB not to modify its output to run on a different Db2 version from the version for which the data was collected.</td>
</tr>
<tr>
<td>partition-rotation</td>
<td>N</td>
<td>Directs DSNADMSB not to check partition rotation.</td>
</tr>
<tr>
<td>output-method</td>
<td>R</td>
<td>R directs DSNADMSB to write output data to the job stream. When output-method is R, output-info must be NONE.</td>
</tr>
<tr>
<td>output-info</td>
<td>NONE</td>
<td>This is the PMR number for the problem that requires data collection.</td>
</tr>
</tbody>
</table>

The JCL for the step that executes DSNADMSB looks like this:

```bash
//DSNADMSB EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//SYSTSPRT DD SYSOUT**
//SYSPRINT DD SYSOUT**,DCB=(RECFM=VB,LRECL=4096)
//SYSDUMP DD SYSOUT**
```
Example: Collecting environment data

Suppose that you want DSNADMSB to retrieve environment data about the ASHEMA.ATABLE user table. You want DSNADMSB to send the output to the job stream.

The parameter values that you specify are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>table-creator</td>
<td>ASHEMA</td>
<td>These two parameters direct DSNADMSB to collect data about table ASHEMA.ANAME.</td>
</tr>
<tr>
<td>table-name</td>
<td>ATABLE</td>
<td></td>
</tr>
<tr>
<td>catalog-creator</td>
<td>DEFAULT</td>
<td>For catalog queries, directs DSNADMSB to use the default catalog table qualifier of SYSIBM.</td>
</tr>
<tr>
<td>plan-info</td>
<td>NONE</td>
<td>This value must be NONE, because PLAN_TABLE data is not being collected.</td>
</tr>
<tr>
<td>collect-dll</td>
<td>Y</td>
<td>Directs DSNADMSB to collect the data definition statements that created ASHEMA.ANAME and associated objects.</td>
</tr>
<tr>
<td>collect-stats</td>
<td>Y</td>
<td>Directs DSNADMSB to collect statistics about tables from the Db2 catalog.</td>
</tr>
<tr>
<td>collect-colstats</td>
<td>N</td>
<td>Directs DSNADMSB to collect statistics about table columns from the Db2 catalog.</td>
</tr>
<tr>
<td>edit-dll</td>
<td>Y</td>
<td>Directs DSNADMSB to modify the data definition language statements that it generates so that the data definition language statements can be more easily executed by IBM Support.</td>
</tr>
<tr>
<td>edit-version-mode</td>
<td>NONE</td>
<td>Directs DSNADMSB not to modify its output to run on a different Db2 version from the version for which the data was collected.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
<td>Explanation</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>partition-rotation</td>
<td>N</td>
<td>Directs DSNADMSB not to check partition rotation.</td>
</tr>
<tr>
<td>output-method</td>
<td>R</td>
<td>R directs DSNADMSB to write output data to the job stream. When output-method is R, output-info must be NONE.</td>
</tr>
<tr>
<td>output-info</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>pmr-info</td>
<td>12345.000.000</td>
<td>This is the PMR number for the problem that requires data collection.</td>
</tr>
</tbody>
</table>

The JCL for the step that executes DSNADMSB looks like this:

```
//DSNADMSB EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//SYSTSPRT DD SYSOUT=*  
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=4096)
//SYSUDUMP DD SYSOUT=* 
//SYSSTIN DD *  
   DSN SYSTEM(DSN)  
   RUN PROGRAM(DSNADMSB) PLAN(DSNADMSB)
//INPUTP DD *  
   ASHEMA  
   ATABLE  
   DEFAULT  
   NONE  
   Y  
   N  
   Y  
   NONE  
   N  
   R  
   NONE  
   12345.000.000
```

**Collecting data for a table with a long table name with embedded blanks**

Suppose that you want to collect the same data as in the previous example for a table with the schema name SYSADM, and the following name, which is 85 bytes long:

A234567891 234567892 234567893 234567894 234567895 234567896 234567897- 234567898 2345

The record length of the INPUTP data set is 80 bytes, but only the first 71 bytes can contain input data, so you need to split the schema name and table name across input records. You can split the table name after any non-blank character in the record. DSNADMSB trims blanks at the end of each record.

The INPUTP data set might look like this. The table name is split across three records, after positions 50 and 65.

```
SYSADM
A234567891 234567892 234567893 234567894 234567895 234567896 234567897- 234567898 2345
```

The JCL for the step that executes DSNADMSB looks like this:

```
//DSNADMSB EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//SYSTSPRT DD SYSOUT=*  
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=4096)
//SYSUDUMP DD SYSOUT=* 
//SYSSTIN DD *  
   DSN SYSTEM(DSN)  
   RUN PROGRAM(DSNADMSB) PLAN(DSNADMSB)
//INPUTP DD *  
   ASHEMA  
   ATABLE  
   DEFAULT  
   NONE  
   Y  
   Y  
   N  
   Y  
   N  
   R  
   N  
   12345.000.000
```
Y
NONE
N
R
NONE
12345.000.000

The first continued line must contain a blank after the plus sign, because the character at position 51 of the table name is a blank. The second continued line must not contain a blank after the plus sign, because the character at position 66 of the table name is not a blank.
Appendix E. DSNTSMFD

The DSNTSMFD program decompresses Db2 trace records that were compressed when they were written to SMF.

Trace records are compressed when subsystem parameter SMFCOMP is set to ON.

**Authorization required**

You need no special authorization is needed to run DSNTSMFD.

**Input**

Input to the DSNTSMFD program is one or more data sets that contain Db2 trace records in standard SMF format. The data sets can contain SMF records of all types, but DSNTSMFD decompresses only SMF type 100, 101, or 102 records.

The input data sets are allocated to DD name SMFINDD.

**Output**

The DSNTSMFD program produces the following output:

- A data set that contains all of the Db2 trace records that are in the input data set. SMF type 100, 101, or 102 records are decompressed in the output data set. If DSNTSMFD cannot decompress the SMF type 100, 101, or 102 records, DSNTSMFD writes the compressed records to the output data set and issues a warning.
  
  This output data set is allocated to DD name SMFOUTDD.

- A data set that contains details about decompression, such as the number of records that were decompressed, and the amount of space that was saved through compression.
  
  This output data set is allocated to DD name SYSPRINT.

**Before running DSNTSMFD**

Certain activities might be required before you run the DSNTSMFD program.

Before running DSNTSMFD:

- Prepare DSNTSMFD for execution.
  
  Customize and run job DSNTEJDS to do that. The job prolog contains instructions for customization.

- Dump SMF data to sequential data sets.
  
  Use a utility such as IFASMFDP to do that.
Data sets that DSNTSMFD uses

The DSNTSMFD utility uses a number of data sets during its operation.

The following table lists the data sets that DSNTSMFD uses.

<table>
<thead>
<tr>
<th>Data set type</th>
<th>DD name</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>SMFINDD</td>
<td>One or more data sets that contain Db2 trace records in standard SMF format. The data sets are sequential data sets that contain the output of an SMF dump utility, such as IFASMFDP.</td>
<td>Yes</td>
</tr>
<tr>
<td>Output</td>
<td>SMFOUTDD</td>
<td>A data set into which DSNTSMFD writes the trace records that are in the input data sets, with SMF type 100, 101, and 102 records decompressed. This data set must have the same data set characteristics as the input data set, but must be larger than the total size of all input data sets, to accommodate the decompressed records.</td>
<td>Yes</td>
</tr>
<tr>
<td>Output</td>
<td>SYSPRINT</td>
<td>A data set into which DSNTSMFD writes a report about SMF record decompression, such as the number of records that were decompressed, and the amount of space that was saved through compression.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Examples of DSNTSMFD invocation

Use the DSNTSMFD invocation examples as models for generating your own DSNTSMFD output.

Example: Decompression of Db2 trace records

Suppose that an SMF data set contains compressed Db2 trace records of SMF type 100, 101, or 102. You have dumped the data into sequential data set DSN1110.SMFDATA. You want to write all of the SMF data to data set DSN1110.SMFOUT, and you want any compressed SMF type 100, 101, or 102 records to be decompressed in DSN1110.SMFOUT.

The JCL for the step that executes DSNTSMFD looks like this:

```
//RUNSMFD EXEC PGM=DSNTSMFD
//SYSPRINT DD SYSOUT=*  
//SYSUDUMP DD SYSOUT=*  
//SMFINDD DD DSN=DSN1110.SMFDATA,  
//          DISP=SHR  
//SMFOUTDD DD DSN=DSN1110.SMFOUT,  
```
// LIKE=DSN1110.SMFDATA,
// DISP=(,CATLG),
// UNIT=SYSDA,
// SPACE=(TRK,(1200,200))

The output looks like this:

```
*** DSNTSMFD *** STARTING 2011/06/27 15:55:39

Total records read:.................................. 146232
Total DB2 records read:............................. 146183
Total DB2 compressed records read:.................. 146183
Total DB2 compressed records decompressed: ....... 146183
Total non-DB2 records read:........................ 49

Aggregate size of all input records:............. 60334251 57M
Aggregate size of all input DB2 records:............ 60323008 57M
Aggregate size of all DB2 compressed records:..... 60323008 57M
Aggregate size of all output DB2 records:......... 102449084 97M
Aggregate size of all DB2 expanded records:...... 102449084 97M
Aggregate size of all non-DB2 input records:...... 11243 10K

Percentage saved using compression.............. 41%

Details by DB2 subsystem
Subsystem ID: DB2A
Number of records:.................................. 146183
Number of compressed records:..................... 146183
Aggregate size of DB2 records:.................... 60323008 57M
Aggregate size of DB2 compressed records:........ 60323008 57M
Aggregate size of DB2 expanded records:.......... 102449084 97M
Percentage saved using compression.............. 41%
```

Appendix E. DSNTSMFD 1121
Appendix F. How real-time statistics are used by Db2 utilities

Db2 utilities use real-time statistics to optimize data processing and operations.

Utilities can use real-time statistics to calculate how data processing is done. This is more efficient than using statistics typically gathered by the RUNSTATS utility or stored in catalogs.

The use of real-time statistics eliminates some of the dependency on regularly running the RUNSTATS utility, which is processing intensive and time consuming.

**Restriction:** Real-time statistics does not generate statistics for certain items in the directory, such as the SYSLGRNX table space and its corresponding indexes DSNLLX01 and DSNLLX02, for example.

Using third-party vendor solutions that do not correctly manage real-time statistics can cause unexpected errors.

If real-time statistics are available and the system parameter UTSORTAL is set to YES, the following utilities use real-time statistics to help determine how data is processed:

- CHECK DATA
- CHECK INDEX
- REBUILD INDEX
- REORG TABLESPACE
- RUNSTATS

Db2 issues messages DSNU3350I and DSNU3351I to indicate the estimated and actual sort quantity for each utility sort task. It also issues DSNU3357I and these values are summarized for each utility invocation when it completes.

The REORG TABLESPACE utility also uses real-time statistics to determine the size of a hash space when reorganizing a hash table space and AUTOESTSPACE YES is specified.

Additionally, the RUNSTATS utility uses real-time statistics when determining the number of records to include when collecting a sampling of statistics.

Db2 issues message DSNU3343I if there are no real-time statistics available. This message can be issued for either table spaces or indexes. When message DSNU3343I is returned, Db2 tries to gather real-time statistics either from associated indexes or table spaces, depending on what kind of real-time statistics were not available. If no real-time statistics are available, Db2 uses RUNSTATS based estimations.

When real-time statistics are not available, and a RUNSTATS control statement with TABLESAMPLE SYSTEM n is run, RUNSTATS issues a message, and continues with TABLESAMPLE SYSTEM AUTO behavior. If real-time statistics are not available when RUNSTATS is run with TABLESAMPLE SYSTEM AUTO, RUNSTATS sets the sampling rate to 100 and continues to run.
Table space and index characteristics

Utilities regularly gather information about table space or index characteristics. The information is used to calculate statistics that help determine how a utility processes data.

Utilities read the totals number of rows from the column TOTALROWS in the table SYSIBM.SYSTABLESPACESTATS and the number of associated index keys from column TOTALENTRIES in the table SYSIBM.SYSINDEXSPACESTATS. The statistics that are calculated from this information are used to estimate the number of records that need to be sorted and the size of the required sort work data sets.

**Recommendation:** To prevent utilities from using incorrect values when table spaces are replaced by utilities such as DSN1COPY or other utilities that are not controlled by Db2, column information can be set to NULL. When information for the columns is set to NULL, the number of records is estimated based on statistics that are gathered by RUNSTATS. The columns are then re-initialized the next time REORG TABLESPACE, LOAD REPLACE, or REBUILD INDEX runs. Alternatively, running RUNSTATS with SHRLEVEL REFERENCE re-initializes the real-time statistics column values.

**Related reference:**

[UT SORT DATA SET ALLOCATION field (UTSORTAL subsystem parameter)](Db2 Installation and Migration)

**Related information:**
Appendix G. Delimited file format

A delimited file is a sequential file with column delimiters. Each delimited file is a stream of records, which consists of fields that are ordered by column.

Each record contains fields for one row. Within each row, individual fields are separated by column delimiters. All fields must be delimited character strings, non-delimited character strings, or external numeric values. Delimited character strings can contain column delimiters and can also contain character string delimiters when two successive character string delimiters are used to represent one character.

All characters in all records are in the same CCSID. If EBCDIC or ASCII data contains DBCS characters, the data must be in an appropriate mixed CCSID. If the data is Unicode it must be in CCSID 1208.

The following figure describes the format of delimited files that can be loaded into or unloaded from tables by using the LOAD and UNLOAD utilities.

```
Delimited file ::= Row 1 data ||
  Row 2 data ||
  .
  .
  .
  Row n data

Row i data ::= Cell value(i,1) || Column delimiter ||
  Cell value(i,2) || Column delimiter ||
  .
  .
  .
  Cell value(i,m)

Column delimiter ::= Character specified by COLDEL option;
  the default value is a comma (,)

Cell value(i,j) ::= Leading spaces ||
  External spaces ||
  External numeric values ||
  Delimited character string ||
  Non-delimited character string ||
  Trailing spaces

Non-delimited character string ::= A set of any characters except
  a column delimiter

Delimited character string ::= A character string delimiter ||
  A set of any characters except a
  character string delimiter unless
  the character string delimiter is
  part of two successive character
  string delimiters ||
  A character string delimiter ||
  Trailing garbage

Character string delimiter ::= Character specified by CHARDEL option; the default
  value is a double quotation mark ("")

Trailing garbage ::= A set of any characters except a column delimiter
```

Related concepts:
**Data types in delimited files**

The LOAD and UNLOAD utilities can process delimited files. When you load a delimited file, LOAD requires that the data in the file be in a certain form depending on the data type. Similarly, when you unload data to a delimited file, UNLOAD writes the data in a certain form depending on the data type.

The following table identifies the acceptable data type forms for the delimited file format that the LOAD and UNLOAD utilities use.

*Table 152. Acceptable data type forms for delimited files*

<table>
<thead>
<tr>
<th>Data type</th>
<th>Acceptable form for loading a delimited file</th>
<th>Form that is created by unloading a delimited file</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR, VARCHAR</td>
<td>A delimited or non-delimited character string</td>
<td>Character data that is enclosed by character delimiters. For VARCHAR, length bytes do not precede the data in the string.</td>
</tr>
<tr>
<td>GRAPHIC (any type)</td>
<td>A delimited or non-delimited character stream</td>
<td>Data that is unloaded as a delimited character string. For VARGRAPHIC, length bytes do not precede the data in the string.</td>
</tr>
<tr>
<td>INTEGER (any type)</td>
<td>A stream of characters that represents a number in EXTERNAL format</td>
<td>Numeric data in external format.</td>
</tr>
<tr>
<td>DECIMAL (any type)</td>
<td>A character string that represents a number in EXTERNAL format</td>
<td>A string of characters that represents a number.</td>
</tr>
<tr>
<td>DECFLOAT EXTERNAL</td>
<td>A character string that represents</td>
<td>A SQL numeric constant.</td>
</tr>
<tr>
<td>FLOAT</td>
<td>A representation of a number in the range -7.2E+75 to 7.2E+75 in EXTERNAL format</td>
<td>A string of characters that represents a number in floating-point notation.</td>
</tr>
<tr>
<td>BLOB, CLOB</td>
<td>A delimited or non-delimited character string</td>
<td>Character data that is enclosed by character delimiters. Length bytes do not precede the data in the string.</td>
</tr>
<tr>
<td>DBCLOB</td>
<td>A delimited or non-delimited character string</td>
<td>Character data that is enclosed by character delimiters. Length bytes do not precede the data in the string.</td>
</tr>
<tr>
<td>DATE</td>
<td>A delimited or non-delimited character string that contains a date value in EXTERNAL format</td>
<td>Character string representation of a date.</td>
</tr>
</tbody>
</table>
Table 152. Acceptable data type forms for delimited files (continued)

<table>
<thead>
<tr>
<th>Data type</th>
<th>Acceptable form for loading a delimited file</th>
<th>Form that is created by unloading a delimited file</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME</td>
<td>A delimited or non-delimited character string that contains a time value in EXTERNAL format</td>
<td>Character string representation of a time.</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>A delimited or non-delimited character string that contains a timestamp value in EXTERNAL format</td>
<td>Character string representation of a timestamp.</td>
</tr>
<tr>
<td>TIMESTAMP WITH TIME ZONE</td>
<td>A delimited or non-delimited character string that contains a timestamp with time zone value in EXTERNAL format</td>
<td>Character string representation of a timestamp with time zone.</td>
</tr>
</tbody>
</table>

Note:
1. Field specifications of INTEGER or SMALLINT are treated as INTEGER EXTERNAL.
2. Field specifications of DECIMAL, DECIMAL Packed, or DECIMAL ZONED are treated as DECIMAL EXTERNAL.
3. Field specifications of FLOAT, REAL, or DOUBLE are treated as FLOAT EXTERNAL.
4. EBCDIC graphic data must be enclosed in shift-out and shift-in characters.

Examples of delimited files

Use the examples as models to specify your own delimited files.

**Example 1: Delimited file with delimited character strings**

The following figure shows an example of a delimited file with delimited character strings. In this example, the column delimiter is a comma (,). Because the character strings contain the column delimiter character, they must be delimited with character string delimiters. In this example, the character string delimiter is a double quotation mark ("").

"Smith, Bob",4973,15.46
"Jones, Bill",12345,16.34
"Williams, Sam",452,193.78

*Figure 117. Example of a delimited file with delimited character strings*

**Example 2: Delimited file with non-delimited character strings**

The following figure shows an example of a delimited file with non-delimited character strings. In this example, the column delimiter is a semicolon (;). Because the character strings do not contain the column delimiter character, they do not need to be delimited with character string delimiters.

Smith, Bob;4973;15.46
Jones, Bill;12345;16.34
Williams, Sam;452;193.78

*Figure 118. Example of a delimited file with non-delimited character strings*
Information resources for Db2 11 for z/OS and related products

Information about Db2 11 for z/OS and products that you might use in conjunction with Db2 11 is available online in IBM Knowledge Center or on library websites.

Obtaining Db2 for z/OS publications

Current Db2 11 for z/OS publications are available from the following websites:

http://www-01.ibm.com/support/docview.wss?uid=swg27039165

Links to IBM Knowledge Center and the PDF version of each publication are provided.

Db2 for z/OS publications are also available for download from the IBM Publications Center [http://www.ibm.com/shop/publications/order].

In addition, books for Db2 for z/OS are available on a CD-ROM that is included with your product shipment:

- Db2 11 for z/OS Licensed Library Collection, LK5T-8882, in English. The CD-ROM contains the collection of books for Db2 11 for z/OS in PDF format. Periodically, IBM refreshes the books on subsequent editions of this CD-ROM.

Installable information center

You can download or order an installable version of the Information Management Software for z/OS Solutions Information Center, which includes information about Db2 11 for z/OS, QMF™, IMS, and many Db2 Tools for z/OS products. You can install this information center on a local system or on an intranet server. For more information, see [http://www-01.ibm.com/support/knowledgecenter/SSEPEK_11.0.0/com.ibm.db2zl1.doc/src/alltoc/installabledzic.html].
Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785 US

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.
Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785 US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as shown below:

© (your company name) (year).
 Portions of this code are derived from IBM Corp. Sample Programs.
 © Copyright IBM Corp. (enter the year or years).

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

This information is intended to help you to use Db2 for z/OS utilities. This information also documents General-use Programming Interface and Associated Guidance Information and Product-sensitive Programming Interface and Associated Guidance Information provided by Db2 11 for z/OS.
General-use Programming Interface and Associated Guidance Information

General-use Programming Interfaces allow the customer to write programs that obtain the services of Db2 11 for z/OS.

General-use Programming Interface and Associated Guidance Information is identified where it occurs by the following markings:

GUPI

Product-sensitive Programming Interface and Associated Guidance Information

Product-sensitive Programming Interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of this IBM software product. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-sensitive Programming Interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

Product-sensitive Programming Interface and Associated Guidance Information is identified where it occurs by the following markings:

PSPI

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered marks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at: [http://www.ibm.com/legal/copytrade.shtml]

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.
Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions:

Applicability: These terms and conditions are in addition to any terms of use for the IBM website.

Personal use: You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use: You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights: Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NONINFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Privacy policy considerations

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies
and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

Glossary

The glossary is available in IBM Knowledge Center.

See the Glossary topic for definitions of Db2 for z/OS terms.
Index

Numerics
32K
  option of DSN1COMP 951
  option of DSN1COPY 966
  option of DSN1PRNT 1012

A
abend
  forcing 210
ABEND
  option of DIAGNOSE 207
Access Method Services, new active log definition 918
accessibility
  keyboard xvii
shortcut keys xviii
ACCESSPATH
  option of MODIFY STATISTICS 401
  option of REORG TABLESPACE 594
ACTION
  option of DSN1SDMP 1025
ACTION2
  option of DSN1SDMP 1026
active
  utility status 39
active log
  adding to BSDS 918
  data set with I/O error, deleting 921
  deleting from BSDS 919
  enlarging 919
  recording from BSDS 919
active logs
  preformatting 901
ADMIN_INFO_SQL
  modifying output 1109
advisory state
  1037
AFTER
  option of DSN1SDMP 1026
AFTER2
  option of DSN1SDMP 1027
AGE
  option of MODIFY STATISTICS 401
ALIAS
  option of DSNJU003 914
aliases
  adding to communication record in BSDS example 926
  removing from communication record in BSDS example 926
ALL
  option of LISTDEF 223
  option of REBUILD INDEX 432
  option of RUNSTATS INDEX 760
ALLDUMPS
  option of DIAGNOSE 205
ARCHIVE
  option of LISTDEF 224
archive log
  adding to BSDS 920
  deleting from BSDS 920
  preformatting 901
ARCHLOG
  option of REPORT 708
ASCII
  option of LOAD 270
  option of UNLOAD 831
audit authoritative authority records 1091
audit authorization failure records 1075
audit DDL create/alter/drop records 1082
audit DDL grant/revoke records 1081
audit DML statement records 1085
audit first read records 1085
audit first write records 1084
authorization ID
  changing to a role 69
namming convention xviii
authorization IDs
  utilities 5
AUTOESTSPACE
  option of REORG TABLESPACE 573
AUX
  option of REORG TABLESPACE 582
AUXERROR INVALIDATE
  option of CHECK DATA 97
AUXERROR REPORT
  option of CHECK DATA 97
AUXERROR, option of CHECK DATA 80
auxiliary CHECK-pending (ACHKP) status
  description 1037
  resetting 98, 1037
auxiliary index
  reorganizing after loading data 360
auxiliary warning (AUXW) status
  description 1038
  resetting 1038

B
BACKOUT
  option of DSNJU003 913
  option of LOAD 254
  option of RECOVER 471
backup
  example of dump class 59
  example of overwriting 58
backup system
  recovering
    example 738
    recovering after database volumes restored
    example 738, 739
BACKUP SYSTEM utility
  authorization 51
  compatibility 56
data sets 55

© Copyright IBM Corp. 1983, 2017
BACKUP SYSTEM utility (continued)
description 51
dumping 56
examples 58
execution phases 51
history, printing 927
options 53
output 51
prerequisite actions 54
restarting 57
syntax 52
terminating 57
backup to tape
example 58
BASE
option of LISTDEF 223
BASIC
option of LISTDEF 224
basic predicate 588
BETWEEN predicate 589
BIGINT
option of LOAD 296
option of UNLOAD FROM TABLE 851
BINARY
option of LOAD 296
option of UNLOAD FROM TABLE 851
BLOB
option of LOAD 302
option of UNLOAD FROM TABLE 856
BOTH
option of RUNSTATS INDEX 762
option of RUNSTATS TABLESPACE 753
BSDS
active log data set status 939
communication records, printing 927
converting 895
data set references, adding and deleting 923
GENERIC LUNAME parameter, updating 916
LOCATION value, updating 913
LUNAME value, updating 913
PASSWORD value, updating 913
updating 903
VSAM catalog name, changing 913
BSDS timestamps
DSNJU004 output 937
BUFNO
option of TEMPLATE 802

C
CANCEL
option of DSNJU003 912
catalog
checking for inaccuracies
example 701
repairing 698
RUNSTATS utility 773
updating 65
CATALOG
option of DSNJU003 909
option of UNLOAD 828
statement of REPAIR utility 688
catalog and directory
comparing DBDs
example 700
creating recovery point of consistency 493
defining high-level qualifier 923
catalog and directory (continued)
deleting rows 393
reorganizing 628
REPORT 709
catalog and directory objects
copying 165
quiescing 420
catalog indexes
rebuilding 452
catalog name
changing for objects 70
catalog SPACE columns
updating
example 791
updating for all storage groups
example 791
updating for several storage groups
example 792
catalog table spaces
corresponding directory table spaces 629
catalog tables
data collected by STOSPACE utility 789
loading data into 305
order of recovering objects 485
outdated information, removing 387
SPACE column 791
SPACEF column 791
statistics history, clearing outdated information 399
statistics, deleting 403
SYSCOPY
deleting rows 393
SYINDEXES
updating space information 790
SYSGRNX
deleting rows 393
SYSTABLESPACE
updating space information 790
updating for installation 69
updating for migration 69
CATENFM utility
authorization 61
compatibility 63
data sets 62
description 61
execution phases 61
options 62
output 61
syntax 61
terminating 64
CATMAINT utility
authorization 65
data sets needed 65
description 65
execution phases 65
options 66
output 65
prerequisite actions 62
restarting 68
syntax 66
terminating 68
CCSID
option of LOAD 270
option of UNLOAD 831
CCSID information, deleting from BSDS 911
CCSID
option of DSNJU003 911
clone table space
repairing
  example 701
cold start
  example, creating a conditional restart control record 921
command record
  adding
    example 925
  adding to BSDS
    example 925
  updating
    example 925
compatibility
  BACKUP SYSTEM 56
  CATENFM 63
  CHECK DATA 90
  CHECK INDEX 111
  CHECK LOB 128
  COPY 150
  COPYTOCOPY 193
  DEFINE NO objects 9
  EXEC SQL 211
  LISTDEF 225
  LOAD 312
  MERGECOPY 382
  MODIFY STATISTICS 402
  OPTIONS 411
  QUIESCE 419
  REBUILD INDEX 443
  RECOVER 473
  REORG INDEX 539
  REORG TABLESPACE 618
  REPAIR 692
  REPORT 709
  RESTORE SYSTEM 737
  RUNSTATS 770
  STOSPACE 789
  TEMPLATE 810
  UNLOAD 823
complete
  option of CATENFM 62
compression
  estimating disk savings 949
  (continued)
  LOB changes 647
  concatenating input records
    LOAD
      example 366
CONCURRENT
  option of COPY 145, 166
  concurrent copies
    COPYTOCOPY utility restriction 185
    invoking 145
    making 166
  conditional copy
    example 181
  conditional restart control record
    creating 911, 921
    example 925
    reading 940
    status printed by DSNJU004 927
  connection-name
    naming convention xviii
CONSTANT
  option of LOAD 289
  option of UNLOAD FROM TABLE 855
CONSTANTIF
  option of LOAD 289
  constraint violations
    checking 73
CONTINUEIF
  option of LOAD 273
CONTROL
  option of DSNU CLIST 29
  control interval
    LOAD REPLACE effect on 361
    LOAD REPLACE, effect of 316
    REORG TABLESPACE, effect of 653
  control statement 18
  Control Statement Data Set Names panel 25
conversion of data types, LOAD utility 340
convert
  option of CATENFM 62
  clones
    clone table data
      example 184
    for local site and recovery site
      example 174
    full image copy
      example 173
    incremental image copy
      example 180
    list of objects
      example 174
    LISTDEF
      example 180
    LOB table spaces
      example 182
    merging
      example 377
      example 386
    merging incremental
      example 385
    parallel processing
      example 177
    templates
      example 179
    updated objects only
      example 184
    copy
      XML schema repository objects 165
COPY
  option of LISTDEF 219
copy consistency 156
copy pool 51
copy pools
description 54
COPY utility
  adding conditional code 168
  allowing other programs to access data 146
  authorization 133
  catalog table, copying 153
  checking pages 143
  compatibility 150
  consistency 156
COPY-pending 162
COPY-pending status, resetting 133
data sets 147
description 133
directory, copying 153
examples 173
execution phases 134
full image copies 153
generation data groups, defining 171
incremental image copy 154
informational COPY-pending status, resetting 133
list of objects 161
MERGECOPY utility, when to use 384
multiple image copies 155
multiple statements 163
naming data sets 155
options 137
output 133
output data sets
  JCL parameters 149
partition, copying 154
performance recommendations 171
prerequisite actions 147
processing in parallel 162
description 142
recovery, preparing for 170
REPORTONLY
  examples 181
restarting 173
restricted states 150
simultaneous jobs 163
syntax 135
terminating 172
COPY-pending status
description 1040
LOAD 356
resetting 356, 1040
COPY1
  option of DSNJU003 908
COPY1VOL
  option of DSNJU003 909
COPY2
  option of DSNJU003 908
COPY2VOL
  option of DSNJU003 909
COPYDDN
  option of COPY 139
  option of COPYTOCOPY 190
  option of LOAD 256, 335
  option of MERGECOPY 380
  option of REORG TABLESPACE 571, 635
COPYDICTIONARY
  option of LOAD 250
COPYDSN
  option of DSNU CLIST 29
COPYDSN2
  option of DSNU CLIST 29
copying cataloged image copy data set
  example 200
copying image copy data set
  example 200
copying last copy
  example 199
copying last full copy
  example 199
copying uncataloged image copy data set
  example 200
COPYTOCOPY utility
  authorization 185
  compatibility 193
copying from specific image copy 195
copying from tape 197
data sets 191
description 185
determining input copy 196
examples 199
execution phases 186
FlashCopy 195
generation data groups, defining 196
incremental image copy 194
JCL parameters 192
lists of objects 188
making copies 194
multiple statements 194
options 188
output 185
output data sets
  size 192
partitions, copying 189
restarting 198
syntax 186
SYSIBM.SYSCOPY records 196
tape mounts, retaining 192
templates, using 196
terminating 198
correlation ID
  naming convention xviii
COUNT
  option of REBUILD INDEX 435
  option of REORG INDEX 529
  option of RUNSTATS TABLESPACE 753, 755
COUNT option
  option of RUNSTATS INDEX 761
CREATE
  option of DSNJU003 911
creating a table
  EXEC SQL example 213
creator
  renaming 69
CRESTART
  option of DSNJU003 911
  cross loader 333
  cross loader function 211
CSRONLY
  option of DSNJU003 913
CURRENT
  option of REPORT 708
current restart 44
CURRENTCOPYONLY
  option of RECOVER 466
cursor
LOAD
  example 373, 374
naming convention xviii

data
  replacing
    example 699
  adding to a table 320
  converting 332
  deleting all 321
DATA
  option of CHECK DATA 76
  option of LOAD 248
  option of REPAIR DUMP 682
  option of REPAIR REPLACE 679
  option of REPAIR VERIFY 679
  option of UNLOAD 827
data compression
  description 330
  dictionary 331
REORG TABLESPACE utility 593
DATA ONLY
  option of BACKUP SYSTEM 53
data set
  name format in ICF catalog 142
  name limitations 811
data set definitions
  changing during REORG 631
Data Set Names panel 24
data set templates
  extent allocations 813
  space calculations 812
data sets
BACKUP SYSTEM 55
CATENFM 62
CATMAINT utility 65
  change log inventory utility (DSNJU003) 903
CHECK DATA 86
CHECK INDEX 108
CHECK LOB 125
concatenating 17
COPY 147
COPYTOCOPY 191
disposition 18
DSN1COPY 972
DSNADMSB 1109
DSNJCNVB utility 895
DSNJCNVT utility 898
DSNTSMD 1120
for copies, naming 155
input 16
LOAD 307
MERGECOPY 381
MODIFY STATISTICS 402
naming convention xviii
online utilities 16
output 16
QUIESCE 418
REBUILD INDEX 440
RECOVER 472
REORG INDEX 535
REORG TABLESPACE 609
REPAIR 691
data sets (continued)
  REPORT 709
  RESTORE SYSTEM 736
  RUNSTATS 767
  security 18
  STOSPACE 788
  UNLOAD 822
data sharing
  backing up group 51
data sharing member
  deactivating
    example 926
data type, specifying with LOAD utility 290
data-only backup
  description 53
  example 58
data-partitioned secondary indexes
  statistics 773
database
  naming convention xviii
DATABASE
  option of LISTDEF 220
  option of REPAIR DBD 685
database copy pool
  recovering
    example 739
DATACLAS
  option of TEMPLATE 802
DATAONLY
  option of DSN1LOGP 993
  option of REPAIR DELETE 681
DataRefresher 332
DATE
  option of MODIFY STATISTICS 402
DATE EXTERNAL
  option of LOAD 299
  option of UNLOAD FROM TABLE 855
Db2 internal format
LOAD 314
Db2 utilities
  packaging 3
  Db2-supplied stored procedures 1035
DB2I
  invoking online utilities 22
    option of DSNU CLIST 29
DBCLOB
  option of LOAD 303
  option of UNLOAD FROM TABLE 857
DBD
  reclaiming space 395
  statement of REPAIR utility 684
DBD01 directory table space
  MERGECOPY restrictions 378, 379
  order of recovering 487
DBETE status
  resetting 1041
DBID
  option of DSN1LOGP 993
  option of DSNJU008 946
  option of REPAIR DBD 685
DBRM (database request module)
  member naming convention xviii
  partitioned data set naming convention xviii
DD name
  naming convention xviii
DDF
  option of DSNJU003 913
DDF Location records 1057
DDNAME option of DSNJU004 929
DEACTIV option of DSNJU003 918
DEADLINE option of REORG INDEX 522
option of REORG TABLESPACE 576
DEC FLOAT option of LOAD 303
option of UNLOAD FROM TABLE 858
DEC FLOAT EXTERNAL option of LOAD 303
option of UNLOAD FROM TABLE 858
DEC FLOAT_ROUNDMODE option of LOAD 274
option of UNLOAD 836
DECIMAL option of UNLOAD FROM TABLE 852
DECIMAL EXTERNAL option of LOAD 298
option of UNLOAD FROM TABLE 854
DECIMAL PACKED option of LOAD 298
option of UNLOAD FROM TABLE 852
DECIMAL ZONED option of the LOAD 298
option of UNLOAD FROM TABLE 853
declared temporary table utility compatibility 7
declaring a cursor EXEC SQL example 213
decompressing SMF trace records DSNTSMFD 1119
DECPT option of LOAD 269
option of UNLOAD 835
DEFAULTIF option of LOAD 303
DEFINE NO objects utility compatibility 9
DEFINE NO table space, loading data 321
DEFINED option of LISTDEF 222
DELAY option of REBUILD INDEX 433
option of REORG INDEX 525
option of REORG TABLESPACE 579
DELETE option of CHECK DATA 82, 95
option of DSNJU003 91
option of MODIFY RECOVERY 390
option of MODIFY STATISTICS 401
statement of REPAIR utility 680
DELETE PROFILE option of RUNSTATS TABLESPACE 750
deleting log data sets with errors 921
DELIMITED option of LOAD 268
option of UNLOAD 834
delimited file format acceptable data types 327
data types 1126
default delimiter values 326, 877
description 1125
delimiters
acceptable data type forms for LOAD and UNLOAD 878
unloading to 877
deleted files
example 365
loading 268, 325
maximum delimiter values 326
delimiters
column 1125
restrictions 877
string 1125
DELMBR option of DSNJU003 917
DESTROY option of DSNJU003 918
DFSMS (Data Facility Storage Management Subsystem) concurrent copies 166
invoking with COPY utility 145
using with DB2 172, 197
DFSMSdss concurrent copy example 181, 182
DFSMSdss COPY operation with utilities refining with subsystem parameters 40
DIAGNOSE option of REPAIR DBD 685
DIAGNOSE utility
ABEND statement description 207
authorization 203
compatibility 208
data sets 207
description 203
DISPLAY statement description 205
examples 208
forcing an abend 210
output 203
restarting 208
syntax 204
terminating 208
WAIT statement description 206
diagnosis
specific type DIAGNOSE example 208
DIR option of TEMPLATE 807
directory MGERECOPY restrictions 378
order of recovering objects 485
RUNSTATS utility 773
disability xvii
DISCARD option of REORG TABLESPACE 602
DISCARDDN option of LOAD 271
option of LOAD PART 284
option of REORG TABLESPACE 600
DISCARDS option of LOAD 272
DISCDSN option of DSNU CLIST 29
DISP option of TEMPLATE 803
DISPLAY
- option of DIAGNOSE  205
DISPLAY UTILITY command
- description 39
RESTORE SYSTEM  737
displaying status of Db2 utilities  39
disposition, data sets  18
DL/I
- loading data  332
DOUBLE
- option of UNLOAD FROM TABLE  855
DRAIN
- option of REORG INDEX  524
- option of REORG TABLESPACE  578
DRAIN_ALLPARTS
- option of REORG TABLESPACE  580
DRAIN_WAIT
- option of CHECK DATA  79
- option of CHECK INDEX  107
- option of CHECK LOB  123
- option of LOAD  275
- option of REBUILD INDEX  434
- option of REORG INDEX  523
- option of REORG TABLESPACE  577
DROP
- option of REPAIR DBD  684
DSN
- option of TEMPLATE  798
DSN1COMP utility
- authorization required  949
- compression estimates  955
- compression savings  955
- data set size, specifying  952
- data sets required  949
DD statements
- SYSPRINT  949
- SYSPUT  949
- description  949
- environment  949
- estimate compression savings  953
- examples  956
- free pages, specifying  952
- free space
  - specifying  953
- full image copy, specifying  953
- LARGE data sets, specifying  952
- maximum number of rows to evaluate  953
- option descriptions  951
- output  958
- page size of input data set, specifying  952
- partitions, specifying number  952
- prerequisite actions  954
- REORG option  955
- syntax  951
DSN1COPY
- copying from image copy  978
DSN1COPY utility
- additional volumes, for SYSUT2  973
- altered table  972
- authorization required  963
- checking validity of input  965
- comparison to DSN1PRNT  970
- copying a table space  970
- copying tables to other subsystems  981
- data set size  967
- data set size, determining  971
- data sets  972
DSN1COPY utility (continued)
- DD statements  972
- description  961
- environment  962
- examples  983
- full image copy  966
- image copy, using as input  978
- inconsistent data
  - checking for  977
- incremental copy  966
- inline copy  967
- LARGE input data set  967
- LOB table space  967
- maximum piece size  968
- OBID translation  969
- OBIDXLAT  971, 977
- option descriptions  965
- output  961
- page size of input data set  966
- page size, determining  971
- partitions, specifying number  967
- prerequisite actions  970
- printing data sets  981
- printing in hexadecimal format  968
- resetting log RBAs  969
- restoring indexes  979
- restoring table spaces  980
- restrictions  963
- scanning input data set for value  969
- segmented table space  966
- syntax  965
DSN1LOGP utility
- archive log data sets on tape, reading  998
- authorization  989
- data changes  993
- data sets required  989
- data sharing example  999
- data sharing requirements  990
- database identifier, using to limit report  994
- DBID  993
- DD statements
  - ACTIVE  990
  - ARCHIVE  990
  - BSDS  990
  - SYSPUR  989
  - SYSPRINT  989
  - SYSSUMRY  989
- description  989
- detail report  1004
- environment  989
- error codes, interpreting  1007
- examples  999
- log data sets, identifying  990
- LUWIDs, reporting on  995
- options  992
- output  1002
- page regression report  1005
- page, limiting report to  994
- RID, using to limit report  995
- summary report  997, 1003
- syntax  991
- SYSCOPY log records  993
- type of log records, limiting report by  996
- unit of recovery identifier, using to limit report  995
- value in log record, limiting report by  997
DSN1PRNT utility
- authorization required  1010
DSN1PRNT utility (continued)
comparing with DSN1COPY utility 1016
data set size 1013
data set size, determining 1016
data sets required 1010
DD statements
SYSPRINT 1010
SYSUT1 1010
description 1009
environment 1009
examples 1017
filtering pages by value 1015
full image copy 1012
incremental copy 1012
inline copy 1012
LARGE data set 1012
LOB table space 1013
number of partitions 1014
options 1011
output 1009
page size 1012
page size, determining 1016
piece size 1013
syntax 1011
SYSUT1 data set 1014
DSN1SDMP utility
action 1025
action, specifying 1026
authorization required 1021
buffers, assigning 1027
DD statements
SDMPIN 1022
SDMPFRNT 1022
SDMPTRAC 1022
SYSSNABEND 1022
SYSSNSTIN 1022
description 1021
dump, generating 1028
environment 1021
examples 1029
options 1023
output 1021
required data sets 1022
selection criteria 1024
syntax 1023
trace destination 1024
traces
modifying 1028
stopping 1028
DSN1SMFP utility
trace data 1051
DSN1SMFP utility
authorization 1053
control statement 1053
dedicated report DDs 1054
environment 1053
examples
extracting, formatting, and printing SMF records 1054
JCL requirements 1053
output 1054
IFCID003 DD 1057
IFCID004 DD 1059
IFCID005 DD 1059
IFCID023 DD 1060
IFCID024 DD 1063
IFCID025 DD 1064
DSN1SMFP utility (continued)
output (continued)
IFCID083 DD 1065
IFCID106 DD 1066
IFCID140 DD 1075
IFCID141 DD 1081
IFCID142 DD 1082
IFCID143 DD 1084
IFCID144 DD 1085
IFCID145 DD 1085
IFCID269 DD 1087
IFCID270 DD 1089
IFCID350 DD 1089
IFCID361 DD 1091
IFCID362 DD 1095
standard report header 1056
SYSPRINT DD 1097
output record structure 1056
prerequisite actions 1053
running 1053
DSNADMSB
authorization 1099
data sets 1109
description 1099
examples 1110
modifying output 1109
output 1099
prerequisite actions 1108
syntax 1100
DSNAME
option of DSNJU003 908
DSNDB01.DBD01
copying restrictions 155
recovery information 709
DSNDB01.SYSCOPYs
copying restrictions 155
DSNDB01.SYSTUTILX
copying restrictions 155
recovery information 709
reinitializing 494
DSNDDB01.SYSTSPCYPY
recovery information 709
DSNDB07
REORG TABLESPACE restriction 553
DSNJCNVB utility
authorization required 895
collection statement 895
data sets 895
MENU 895
data sets 895
description 895
dual BSDSs, converting 895
default environment 895
elements 895
output 896
prerequisite actions 895
running 895
DSNJCNVT utility
authorization required 897
collection statement 898
data sets 898
DD names 898
SYSUT1 898
SYSUT2 898
DSNJCNVT utility (continued)
  DD names (continued)
      SYSUT3  898
      SYSUT4  898
      description  897
      environment  897
      example  898
      output  899
      running  898

DSNJLOGF utility
  data sets required  901
  DD names
      SYSPRINT  901
      SYSUTx  901
  description  901
  environment  901
  example  902
  output  902

DSNJU003 utility
  active logs
      adding  918
      changing  918
      deleting  919
      enlarging  919
      recording  919
  archive logs
      adding  920
      deleting  920
  authorization required  903
  BSDS timestamp field, updating  903
  changing log data sets in the BSDS  923
  control statements
      data sets  903
      cataloging  909
      declaring  907
  DD names
      SYSIN  903
      SYSPRINT  903
      SYSUT1  903
      SYSUT2  903
  description  903
  environment  903
  examples  925
      changing high-level qualifier  923
      NEWCAT statement  923
      options  907
  renaming active log data sets  924
  renaming system data sets  924
  running  904
  syntax  904
  updating dual copy BSDSs  903

DSNJU004 utility
  authorization required  927
  data sets  927
  DD names
      GROUP  927
      MnnBSDS  927
      SYSIN  927
      SYSPRINT  927
      SYSUT1  927
      SYSUT2  927
  description  927
  environment  927
  example  929
  example output  940
  example output with checkpoints  940
  options  929
  output  929

DSNJU004 utility (continued)
  recommendations  928
  running  928
  syntax  929

DSNJU008 utility
  description  945
  example  948
  options  946
  syntax  946

DSNTIJIC job
  copy catalog and directory objects  165

DSNTSMFD
  data sets  1120
  description  1119
  example  1120
  output  1119
  prerequisite actions  1119

DSNTYPE
  option of TEMPLATE  807

DSNU CLIST command
  invoking utilities  26
  options  28
  output  32
  syntax  28

DSNU473I  422

DSNUM
  option of COPY  141
  option of COPYTOCOPY  189
  option of MERGECOPY  379
  option of MODIFY RECOVERY  389
  option of RECOVER  464
  option of REPORT  707

DSNUPROC JCL procedure
  description  33
  options  34
  sample  35

DSSIZE
  option of DSN1COMP  952
  option of DSN1COPY  967
  option of DSN1PRNT  1013
  dump
      generating  1028

DUMP
  option of BACKUP SYSTEM  53
  option of DSNJU008  947
  statement of REPAIR utility  681

DUMPCLASS
  option of BACKUP SYSTEM  53
  dumping copy to tape
      BACKUP SYSTEM  56

DUMPONLY
  option of BACKUP SYSTEM  54

E

EATTR
  option of TEMPLATE  808

EBCDIC
  option of LOAD  270
  option of UNLOAD  831

EDIT
  option of DSNU CLIST  30
  edit routine
      LOAD  239
  enabling-new-function mode
      stopping processing  64
encrypted data
running utilities on 11
END
option of DIAGNOSE 207
END FCINCREMENT
option of BACKUP SYSTEM 53
ENLRSN
option of DSNJU003 910
ENDRBA
option of DSNJU003 909
ENDTIME
option of DSNJU003 911
ENFMON
option of CATENFM 62
ENFORCE
option of LOAD 270, 329
option of RECOVER 470
environment data
collecting
DSNADMSB example 1115, 1116
ERRDDN
option of CHECK DATA 84
option of LOAD 271
error data set
CHECK DATA 84
error range
recovery 483
ERROR RANGE
option of RECOVER 470
ESCAPE clause 591
ESTABLISH FCINCREMENT
option of BACKUP SYSTEM 53
EVENT
option of OPTIONS 409
examples
BACKUP SYSTEM 58
CHECK DATA 99
CHECK INDEX 119
CHECK LOB 130
exception table
auxiliary columns 93
columns 92
creating 92
definition 87
example 101
exception tables
example 99
EXCEPTIONS
option of CHECK DATA 83
option of CHECK LOB 124
exceptions for CHECK DATA
example 101
exceptions, specifying the maximum number
CHECK DATA 83
CHECK LOB 124
EXCLUDE
option of LISTDEF 217, 227
EXEC SQL utility
authorization 211
compatibility 211
cursors 212
declare cursor statement 212
description 211
dynamic SQL statements 212
descriptions 212
examples 213
execution phase 211
output 211
EXEC SQL utility (continued)
starting 213
syntax 211
terminating 213
EXEC statement
description 36
executing
utilities
by creating JCL 36
by using DSNUPROC 33
utilities, DB2I 22
exit procedure
LOAD 347
EXPDL
option of TEMPLATE 803
EXTENDED
option of LISTDEF 225
EXTNDICT
option of DSN1COMP 953
extracted key, calculating, LOAD utility 310
F
fallback recovery 508, 606
FAROFFPOSF column of SYSINDEXPART catalog table
catalog query to retrieve value for 623
FLASHCOPY
option of REORG INDEX 527
option of REORG TABLESPACE 583
FCCOPYDDN
option of COPY 145
option of REORG INDEX 533
option of REORG TABLESPACE 573
field positions
LOAD
example 362
field procedure
LOAD 347
field specification errors
UNLOAD 881
FILTER
option of DSN1LOGP 998
option of DSN1SDMP 1026
filter data set, determining size 149
FILTERDDN
option of COPY 145
FlashCopy
COPYTOCOPY 195
creating with LOAD 335
creating with REBUILD INDEX 453
creating with REORG INDEX 544
creating with REORG TABLESPACE 635
during REORG INDEX
example 551
recovering with 481
FLASHCOPY
option of COPY 144
option of REBUILD INDEX 437
option of REORG INDEX 531
option of REORG TABLESPACE 572
FlashCopy image copies
description 157
FlashCopy relationship
with BACKUP SYSTEM 53
FLOAT
option of LOAD 269, 298
option of UNLOAD 835
FLOAT (continued)
  option of UNLOAD FROM TABLE  854
FLOAT EXTERNAL
  option of LOAD  299
FOR
  option of DSN1SDMP  1026
FOR EXCEPTION
  option of CHECK DATA  82
FOR2
  option of DSN1SDMP  1027
FORCE
  option of BACKUP SYSTEM  53
  option of REORG INDEX  527
  option of REORG TABLESPACE  581
FORCEROLLUP
  option of REBUILD INDEX  435
  option of REORG INDEX  529
  option of REORG TABLESPACE  594
  option of RUNSTATS INDEX  764
  option of RUNSTATS TABLESPACE  758
forcing dump
  DIAGNOSE example  208
Forcing RC 0
  OPTIONS example  413
FORMAT
  option of DSN1PRNT  1015
  option of LOAD  266
FORMAT INTERNAL
  LOAD  314
  option of LOAD  267
  option of UNLOAD  833
FORMAT SQL/DS
  option of LOAD  266
FORMAT UNLOAD
  option of LOAD  266
FORWARD
  option of DSNJU003  912
free space
  LOAD  346
  REORG INDEX utility  548
FREEPAGE
  option of DSN1COMP  952
FREQVAL
  option of REBUILD INDEX  435
  option of REORG INDEX  529
  option of REORG TABLESPACE  594
  option of RUNSTATS INDEX  761
  option of RUNSTATS TABLESPACE  753, 755
FROM TABLE
  option of REORG TABLESPACE  587
  option of UNLOAD  838
FROMCOPY
  option of COPYTOCOPY  190
  option of the COPYTOCOPY  195
  option of UNLOAD  828, 869
FROMCOPYDDN
  option of UNLOAD  829, 869
FROMDUMP
  option of RESTORE SYSTEM  733
FROMLASTCOPY
  option of COPYTOCOPY  189
FROMLASTFLASHCOPY
  option of COPYTOCOPY  190
FROMLASTFULLCOPY
  option of COPYTOCOPY  189
FROMLASTINCRCOPY
  option of COPYTOCOPY  189
FROMSEQNO
  option of UNLOAD  828
FROMVOLUME
  option of COPYTOCOPY  190
  option of UNLOAD  828
FULL
  option of BACKUP SYSTEM  53
  option of COPY  139
full backup
  description  53
  example  58
full image copies
  merging with increment image copies  380
FULLCOPY
  option of DSN1COMP  953
  option of DSN1COPY  966
  option of DSN1PRNT  1012
G
GDGLIMIT
  option of TEMPLATE  803
GDGs
  example  183
  defining  171, 196
genereal-use programming information, described  1133
generation data groups
  defining  171, 196
  using with conditional copy  169
GENERIC
  option of DSNJU003  916
GRAPHIC
  option of LOAD utility  294
  option of UNLOAD FROM TABLE  849
GRAPHIC EXTERNAL
  option of LOAD  294
  option of UNLOAD FROM TABLE  849
GRECP
  description  1042
group buffer pool RECOVER-pending (GRECP) status
  description  1042
  resetting  1042
GRPIPV4
  option of DSNJU003  916
GRPIPV6
  option of DSNJU003  916
GUPI symbols  1133

H
HALT
  option of OPTIONS  409
HEADER
  option of UNLOAD FROM TABLE  843
hexadecimal-constant
  naming convention  xviii
hexadecimal-string
  naming convention  xviii
HIGHRBA
  option of DSNJU003  917
HISTOGRAM
  option of REBUILD INDEX  435
  option of REORG INDEX  529
  option of REORG TABLESPACE  594
  option of RUNSTATS INDEX  762
HISTOGRAM (continued)
  option of RUNSTATS TABLESPACE 754, 755
HISTORY
  option of LISTDEF 224
  option of REBUILD INDEX 435
  option of REORG INDEX 529
  option of REORG TABLESPACE utility 594
  option of RUNSTATS INDEX 763
  option of RUNSTATS TABLESPACE 757
HISTORY ACCESSPATH
  option of RUNSTATS TABLESPACE 758

ICBACKUP column
  SYSBM.SYSCOPY 155
ICUNIT column
  SYSBM.SYSCOPY 155
identify end records 1065
identity columns, loading 318
IFCID003 DD 1057
IFCID004 DD 1059
IFCID005 DD 1059
IFCID023 DD 1060
IFCID024 DD 1063
IFCID025 DD 1064
IFCID083 DD 1065
IFCID106 DD
  application programming defaults 1074
data sharing parameters 1073
DDF parameters 1072
description 1066
installation parameters 1067
  system initialization parameters 1066
IFCID140 DD 1075
IFCID141 DD 1081
IFCID142 DD 1082
IFCID143 DD 1084
IFCID144 DD 1085
IFCID145 DD 1085
IFCID269 DD 1087
IFCID270 DD 1089
IFCID350 DD 1089
IFCID361 DD 1091
IFCID362 DD 1095
IGNORE
  option of LOAD 273
IGNOREFIELDS
  option of LOAD 281
image copies
  unloading data from 868
image copy
  cataloging 149, 192
correctional 168
copying 185
copying from 978
copying specific 195
creating 133
data sets 169
deleting all 394
FlashCopy 157
full 133, 153
making 139
incremental 133
  conditions 156
  COPY 154
  COPYTOCOPY 194
image copy (continued)
  incremental (continued)
    performance advantage 154
    list of objects 161
    making after loading a table 356
    merging 377
    multiple, creating 155
    obtaining information about 168
    on tape 172, 197
image copy data set
  determining size 148
IMPLICIT_TZ
  option of LOAD 277
  option of UNLOAD 837
IMS DPROP 332
IN predicate 591
INCLUDE
  option of LISTDEF 217, 226
  inconsistent data indicator, resetting 679
INCRCOPY
  option of DSN1COPY 966
  option of DSN1PRINTF 1012
INCURSOR
  option of LOAD 249
  option of LOAD PART 285
INDDN
  option of LOAD 248
  option of LOAD PART 284
index
  building during LOAD 343
  checking 103
  checking after loading table 359
  determining when to reorganize 541
  inline statistics while reorganizing
    example 550
  naming convention xix
  rebuilding 427
  rebuilding in parallel 447
  recoverability after rebuilding 452
  reorganizing
    example 549
  restrictive status
    resetting 695
  update statistics while reorganizing
    example 550
INDEX
  option of COPY 138
  option of COPYTOCOPY 188
  option of LISTDEF 221
  option of MODIFY STATISTICS 401
  option of REPAIR 674
  option of REPAIR LEVELID 686
  option of REPAIR SET 683
  option of REPAIR VERSIONS 687
  option of REPORT 706
  option of RUNSTATS TABLESPACE 754
INDEX ALL
  option of REPORT 707
INDEX NONE
  option of REPORT 707
index partitions, rebuilding 445
index space
  recovering 427
index statistics
  updating
    example 783, 786
INVALIDCACHE
option of REBUILD INDEX 435
option of REORG INDEX 529
option of REORG TABLESPACE 594
invalidated packages
identifying 71
Invalidating statements in the dynamic statement cache
example 786
IPNAME
option of DSNJU003 915
IPV4
option of DSNJU003 915
IPV6
option of DSNJU003 915
ISPF (Interactive System Productivity Facility)
Control Statement Data Set Names panel 25
Data Set Names panel 24
utilities panel 23
ITEMERROR
option of OPTIONS 409
J
JCL PARM statement 408
K
KEEPDICTIONARY
example 331
option of LOAD 263, 331
option of LOAD PART 284
option of REORG TABLESPACE 593
key
calculating, LOAD utility 310
foreign, LOAD operation 329
primary, LOAD operation 328
KEY
option of OPTIONS 410
option of REPAIR LOCATE TABLESPACE 676
KEYCARD
option of REBUILD INDEX 435
option of REORG INDEX 529
option of REORG TABLESPACE 594
option of RUNSTATS INDEX 761
option of RUNSTATS TABLESPACE 754
L
LARGE
option of DSN1COMP 952
option of DSN1COPY 967
option of DSN1PRNT 1012
LEAFDISTLIMIT
option of REORG INDEX 527
LEAFLIM
option of DSN1COMP 954
LEAST
option of RUNSTATS INDEX 761
option of RUNSTATS TABLESPACE 753
LENGTH
option of REPAIR DUMP 681
level identifier, resetting 686
LEVELID
statement of REPAIR utility 686
LIB
option of DSNU CLIST 32
LIB (continued)
  option of DSNUPROC 34
library of LISTDEF statements 230
LIKE predicate 591
LIMIT
  option of DSNJU008 948
  option of TEMPLATE 805
  option of UNLOAD FROM TABLE 845
links
  non-IBM Web sites 1134
LIST
  option of CHECK INDEX 105
  option of COPY 137
  option of COPYTOCOPY 188
  option of LISTDEF 219
  option of MERGECOPY 379
  option of MODIFY RECOVERY 389
  option of MODIFY STATISTICS 400
  option of QUIESCE 417
  option of REBUILD INDEX 432
  option of RECOVER 463
  option of REORG INDEX 520
  option of REORG TABLESPACE 565
  option of REPORT INDEX 707
  option of REPORT INDEXSPACE 706
  option of REPORT TABLESPACE 706
  option of RUNSTATS INDEX 760
  option of RUNSTATS TABLESPACE 749
  option of UNLOAD 829
LISTDEF
  objects
    excluding 227
    including 226
LISTDEF library
  example 413
LISTDEF library, specifying 411
LISTDEF utility
authorization 215
catalog and directory objects, specifying 229
CHECK INDEX
  example 120
  compatibility 225
  control statement 226
  placement 231
COPY NO indexes, specifying 219
COPY YES indexes, specifying 219
description 215
examples 234
  using LIST 231
execution phases 215
indexes, specifying 218
LOB indicator keywords 223
LOB objects, including 223
options 217
OPTIONS 233
output 215
partitions, specifying 221
pattern-matching expressions 228
previewing 230
RECOVER
  example 513
retrieving lists 231
restoring 234
restrictions 228
statement library 230
syntax 215
templates, using 233
LISTDEF utility (continued)
  terminating 234
LISTDEFDD
  option of OPTIONS 409
LISTPARTS
  option of REORG TABLESPACE 566
lists
  adding related objects 227
  COPYTOCOPY
    example 201
    creating 215
    defining
      examples 234
    objects
      excluding 227
      including 226
    previewing 230
    processing order 232
    restarting utilities with 49
    using with other utilities 231
LOAD INTO PART 321
LOAD INTO TABLE
  options 280
LOAD statements
  generated by UNLOAD 880
LOAD utility
  actions after running 355
  adding data 320
  authorization 239
  BINARYXML 303
  building indexes 343
  catalog tables 305
  CHECK-pending (CHKP) status 357
  compatibility 312
  compressing data 330
  COPY-pending status 356
  cross loader 333
cursors 306
  identifying 285
data sets 307
data type compatibility 340
data type conversion 340
data type, specifying 290
data with referential constraints 328
default values, setting criteria for 303
defects, calculating number 311
DEFINE NO table space 321
delimited file format 325
description 239
discarded rows, statistics 351
duplicate keys, effects 315
effects 360
ENFORCE NO
  consequences 329
  enforcing constraints 270
  error, calculating 311
  examples 362
  EXEC SQL statements 333
  exit procedure 347
  failed
    recovering data 360
  field names, specifying 287
  field specifications 287
FlashCopy 335
foreign keys
  calculating 311
LOAD utility (continued)

foreign keys (continued)
  invalid values 329
free space 346
identity columns 318
informational referential constraints 239
inline copies 355
inline copy 352
inline COPY 334
inline statistics 351
input data set, specifying 248
input data, preparing 305
input fields, specifying 342
into-table spec 277
keys
  calculating 310
limiting logging 350
loading data from DL/I 332
LOB column 348
LOB table space
  logging 349
logging 264
map, calculating 311
multilevel security restriction on REPLACE option 240
multiple tables, loading 277
null values, setting criteria for 304
options 248
ordering records 315
output 239
parallel index build 343
partitions
  loading 282, 321
performance recommendations 336
prerequisite actions 305
primary key
  duplicate values 328
  missing values 329
  REBUILD-pending status 346
  resetting 356
RECOVER-pending status 346
recycling version numbers 360
REORG-pending status 346
reorganizing auxiliary index 360
REPLACE
  restrictive states 316
replacing data 255
restarting 353
restricted status 355
row change timestamp columns 318
row selection criteria 285
ROWID columns 318, 347
skipping fields 281
SORTKEYS NO 314
syntax 242
temporal table columns 318
terminating 352
Unicode data 270
variable-length data 315
work data sets
  estimating size 309
XML column 349
XML table space
  logging 350
loading
catalog tables 305
data
cursors 306
loading (continued)
data (continued)
DL/I 332
dynamic SQL 333
  generated by REORG UNLOAD EXTERNAL 318
  generated by UNLOAD 318
  large amounts 229
partition-by-growth 324
partitions 321
  referential constraints 328
variable-length data 315
XML data 324
loading different data types
example 364
loading multiple tables
example 363
loading null values
example 366
loading selected records
example 363
LOB
LOAD
  example 374
  option of DSN1COMP 954
  option of DSN1COPY 967
  option of DSN1PRNT 1013
  option of LISTDEF 223
LOB (large object)
  checking 81
  invalid 98
  missing 98
  orphan 97
  out-of-synch 98
  recovering 494
LOB changes, compression 647
LOB column
  checking data 85
  errors 97
  loading 348
LOB data
  unloading 865
  unloading to spanned record format 866
LOB table space
  copying 185
LOAD LOG 349
REORG LOG 349
reorganizing 644
LOB violations
  resolving 129
LOBERROR
  option of CHECK DATA 81
LOBERROR INVALDATE
  option of CHECK DATA 97
LOBERROR REPORT
  option of CHECK DATA 97
LOBs
  example of checking 101
  local backup copy
COPYTOCOPY
  example 199
LOCALSITE
  option of RECOVER 471
  option of REPORT 708
LOCATE DELETE statement of REPAIR utility 680
LOCATE DUMP statement of REPAIR utility 681
LOCATE INDEX statement of REPAIR utility 677
LOCATE INDEXSPACE statement of REPAIR utility 677
LOCATE REPLACE statement of REPAIR utility  679
LOCATE TABLESPACE statement of REPAIR utility  675
LOCATE VERIFY statement of REPAIR utility  678
LOCATION
   option of DSNJU003   913
location name
   naming convention xix
log
   active
      data set status   939
      printing available data sets   927
   backward recovery   913
   command history, printing   927
   data set
      printing names   927
   forward recovery   912
   record structure, types   996
LOG
   option of CHECK DATA   83
   option of LOAD   264
   option of REORG TABLESPACE   570
   option of REPAIR   674
log copy pools
   system-level backups   57
log data sets
   active, renaming   924
   archive, renaming   924
log data sets with errors, deleting   921
logical partition, checking   113
logical unit name
   naming convention xix
LOGONLY
   option of RECOVER   468
   option of RESTORE SYSTEM   733
LOGRANGES
   option of RECOVER   471
   option of REORG INDEX   525
   option of REORG TABLESPACE   579
LONGLOG
   option of REBUILD INDEX   433
   option of REORG INDEX   524
   option of REORG TABLESPACE   578
LPL status   1037
LRECL
   option of TEMPLATE   802
LRSENDR
   option of DSN1LOGP   993
LRSNSTART
   option of DSN1LOGP   992
LUNAME
   option of DSNJU003   916
LUWID
   option of DSN1LOGP   995

M
MAP
   option of REPAIR DUMP   682
MAPDDN
   option of LOAD   271
mapping table
   REORG TABLESPACE
      example   664
MAPPINGDATABASE
   option of REORG TABLESPACE   577
MAPPINGTABLE
   option of REORG TABLESPACE   577
MAXERR
   option of UNLOAD   835
MAXPRIME
   option of TEMPLATE   807
MAXRO
   option of REBUILD INDEX   433
   option of REORG INDEX   523
   option of REORG TABLESPACE   577
MAXROWS
   option of DSN1COMP   953
   media failure
      resolving   130
MEMBER
   option of DSNJU004   929
   member name
      naming convention xix
MEMBERID
   option of DSNJU003   918
MEPLs
   displaying
      DIAGNOSE example   208
MERGECOPY utility
   inline copies   383
MERGECOPY utility
   authorization   377
   compatibility   382
   COPY utility, when to use   384
   data sets   381
   DBD01   378, 379
   description   377
   different types of copies, merging restrictions   383
   directory table spaces   378
   examples   385
   lists, using   379
   log information, deleting   384
   LOG RBA inconsistencies, avoiding   384
   options   379
   output   377
   partitions, merging copies   379
   phases of execution   378
   restarting   384
   restrictions   377
   syntax   378
   SYSDBDXA   378, 379
   SYSTSCPY   378, 379
   SYSUTILX   378, 379
   terminating   384
   type of copy   382
   which copies are used   383
   merging copies of data sets   383
MESSAGE
   option of DIAGNOSE   207
MGMTCLAS
   option of TEMPLATE   802
   missing LOB   98
MIXED
   option of LOAD   303
MODELDCB
   option of TEMPLATE   802
MODIFY RECOVERY utility
   authorization   388
   compatibility   393
   copies, deleting   394
   data sets   392
   DBD, reclaiming space   395
   description   387
   examples   397
MODIFY RECOVERY utility (continued)
  lists
    example 397  
  output 387  
  phases of execution 388  
  prerequisites 392  
  recycling version numbers 396  
  REORG after adding column, improving performance 395  
  restarting 393  
  syntax 389  
  SYSCOPY, deleting rows 393  
  SYSLGRNX, deleting rows 393  
  terminating 393  
MODIFY STATISTICS utility
  authorization 399  
  compatibility 402  
  data sets 402  
  description 399  
  examples 404  
  lists, using 400  
  options 400  
  output 399  
  phases of execution 399  
  restarting 403  
  statistics history, deleting 403  
  syntax 400  
  terminating 403  
monitoring
  utility status 39  
MOST
  option of RUNSTATS INDEX 761  
  option of RUNSTATS TABLESPACE 753  
multilevel security with row-level granularity
  authorization restrictions for utilities 5  
  LOAD REPLACE authorization restrictions 240  
  REORG TABLESPACE authorization restrictions 555  
  UNLOAD authorization restrictions 821  
N
naming conventions xvii  
NBRSECND
  option of TEMPLATE 807  
NEAROFFPOSF column of SYSINDEXPART catalog table
  catalog query to retrieve value for 623  
  new-function mode, converting to 63  
NEWCAT
  option of DSNJU003 913  
NEWCOPY
  option of MERGECOPY 380  
NEWLOG
  option of DSNJU003 907  
NEWMAXRO
  option of REORG INDEX 526  
  option of REORG TABLESPACE 581  
NGENERIC
  option of DSNJU003 916  
NGRPV4
  option of DSNJU003 917  
NGRPV6
  option of DSNJU003 917  
NOALIAS
  option of DSNJU003 917  
NOAREORPEND
  option of REPAIR SET 684  
NOAREORPENDSTAR
  option of REPAIR SET 684  
NOAUXCHKP
  option of REPAIR SET 684  
NOAUXWARN
  option of REPAIR SET 684  
NOCHECKPAGE
  option of COPY 143  
NOCHECKPEND
  option of LOAD 271  
  option of REPAIR SET 683  
NOCOPYPEND
  option of LOAD 265  
  option of MODIFY RECOVERY 391  
  option of REPAIR SET 683  
NODUMPS
  option of DIAGNOSE 205  
NOIPNAME
  option of DSNJU003 917  
NOIPV4
  option of DSNJU003 917  
NOIPV6
  option of DSNJU003 917  
NOLUNAME
  option of DSNJU003 917  
  nonindexed row
    example 699  
NOP
  option of REORG TABLESPACE 586  
  option of UNLOAD 832  
NOPASSWD
  option of DSNJU003 916  
NOPRO
  option of REPAIR SET 684  
NORBDPEND
  option of REPAIR SET 683  
NORCVRPEND
  option of REPAIR SET 683  
NOSUBS
  option of LOAD 270  
  option of UNLOAD 832  
NOSYSCOPY
  option of RECOVER 469  
NOSYSREC
  option of REORG TABLESPACE 571  
NOT LOGGED table spaces
  LOAD 361  
  recovering 484  
  REORG TABLESPACE 653  
  not sign, problems with 589  
NULL predicate 593  
null values
  LOAD
    example 366  
NULLIF
  option of LOAD 304  
NUMCOLS
  option of REBUILD INDEX 435  
  option of REORG INDEX 529  
  option of RUNSTATS INDEX 761  
  option of RUNSTATS TABLESPACE 755  
NUMPARTS
  option of DSNCOMP 952  
  option of DSNICOPY 967  
  option of DSN1PRINT 1014  
NUMQUANTILES
  option of REORG TABLESPACE 594  
  option of RUNSTATS INDEX 762  

1156  Utility Guide and Reference
NUMQUANTILES (continued) 754, 755
NUMRECS 281, 363

O

OBID 994
OBIDs 977
OBIDXLAT 969
OBJECT 674
object lists 215
object status 1037
OFF 410
OFF, option of OPTIONS statement 410
OFFLRBA 917
OFFPOSILIMIT 584
OFFSET 997
option of REPAIR DUMP 681
option of REPAIR REPLACE 679
option of REPAIR VERIFY 678
OLDEST_VERSION column 396
online utilities 15
option of LOAD 258
FLASHCOPY 257
OPTIONS utility 412
altering return codes 412
authorization 407
compatibility 411
description 407
effects, handling 409
examples 412
execution phases 407
LISTDEF 233
LISTDEF definition library, specifying 409
options 408
output 407
PREVIEW with LISTDEF 408
PREVIEW with TEMPLATE 408
restarting 412
syntax 407
TEMPLATE definition library, specifying 409
terminating 412
order of records 315
orphan LOB 97
out-of-space condition 199
restarting utilities 48
out-of-synch LOB 98
OUTDDN 685
output fields 871, 873
OVERRIDE 275
owner
renaming 69
OWNER FROM CATMAINT 67
ownership of objects
changing from an authorization ID to a role 69

P

page checking 143
damaged, repairing 695
recovering 483
PAGE 994
option of RECOVER 465
option of REPAIR LOCATE INDEX 678
option of REPAIR LOCATE TABLESPACE 676
page set REBUILD-pending (PSRBD) status 452
description 1043
resetting 1044
PAGES 682
PAGESIZE 952
option of DSN1COMP 952
option of DSN1COPY 956
option of DSN1PRNT 1012
panel
Control Statement Data Set Names 25
Data Set Names 24
DB2 Utilities 23
PARALLEL 108
option of CHECK INDEX 108
option of CHECK DATA 142
option of COPY 251
example 375
option of RECOVER 466
option of REORG INDEX 521
option of REORG TABLESPACE 566
option of UNLOAD 837
parallel index build 447
LOAD 368
example 368
PART 76
option of CHECK DATA 105
option of DSN1008 947
option of LOAD 282, 321
option of QUIESCE 417
option of REBUILD INDEX 432
option of REORG INDEX 521
option of REORG TABLESPACE 568
option of REPAIR 675
option of REPAIR LEVELID 686
option of REPAIR LOCATE INDEX 678
option of REPAIR LOCATE TABLESPACE 676
option of REPAIR SET 683
option of REPAIR VERSIONS 687
option of RUNSTATS INDEX 760
option of RUNSTATS TABLESPACE 749, 754
option of UNLOAD 828
partition
copying 154
partition parallelism
LOAD
example 372
partition-by-growth table space
loading 324
partition-by-growth table spaces
copying 164
rebuilding indexes 446
reorganizing 642
partitioned table space
loading 321
unloading 864
updating statistics 772
partitioned table spaces
reorganizing 642
partitions
rebalancing with REORG 633
rebuilding 633
unloading 864
PARTLEVEL
option of LISTDEF 221
PASSWORD
option of DSNJU003 916
PATH
option of TEMPLATE 808
pattern-matching characters
LISTDEF 229
PCTFREE, option of DSN1COMP 953
PCTPRIME
option of TEMPLATE 807
pending definition changes
reorganizing with 646
performance
affected by
1/O activity 623
COPY 171
LOAD 336
REBUILD INDEX 446
RECOVER 505
REORG INDEX 545
REORG TABLESPACE 395, 637
Persistent Read Only (PRO) restricted status
description 1043
resetting 1043
phase restart 44
phases of execution
BACKUP SYSTEM 51
CATMAINT 65
CHECK DATA 74
CHECK INDEX 104
CHECK LOB utility 121
COPY utility 134
COPYTOCOPY utility 186
EXEC SQL 211
LISTDEF 215
LOAD 240
MERGECOPY 378
MODIFY RECOVERY 388
MODIFY STATISTICS 399
OPTIONS 407
QUIESCE 416
REBUILD INDEX 427
RECOVER 460
REORG INDEX 516
REORG TABLESPACE 555
REPAIR 670
REPORT 704
phases of execution (continued)
RESTORE SYSTEM 732
RUNSTATS 743
STOSPACE 787
TEMPLATE 793
UNLOAD 822
utilities
CATENFM 61
PIECESIZ
option of DSN1COPY 968
option of DSN1PRNT 1013
plan table
retrieving data from
DSNADMSB example 1110, 1112
point-in-time recovery
backout 497
for catalog and directory objects 492
performing 495
PORT
option of DSNJU003 914
POSITION
option of LOAD 289
option of UNLOAD FROM TABLE 845
predicate
basic 588
BETWEEN 589
IN 591
LIKE 591
NULL 593
PREFORMAT
option of LOAD 249
option of LOAD PART 282
option of REORG INDEX 531
option of REORG TABLESPACE 601
preformatting active logs
eexample 902
PRESERVE WHITESPACE
option of LOAD 291, 293
PRESORTED
option of LOAD 250
PREVIEW
option of OPTIONS 408
with LISTDEF utility 230
preview mode
executing utilities 411
PREVIEW mode, executing utilities in 812
previewing lists
eexample 412
previewing templates
eexample 412
PRINT
option of DSN1COPY 968
option of DSN1PRNT 1014
privileges
utilities 5
PRO
option of REPAIR SET 684
processor time
by utilities
monitoring 40
product-sensitive programming information, described 1133
programming interface information, described 1133
PSID
determining 998
option of DSNJU008 946
PSPI symbols 1133
PSRBDPEND
  option of REPAIR SET 684
PUNCHDDN
  option of CHECK DATA  84
  option of CHECK LOB  124
  option of REORG TABLESPACE 599
  option of UNLOAD  829
PUNCHDSN
  option of DSNU CLIST  30

Q
qualifier-name
  naming convention xix
quiesce point
  clone objects
    example  425
  establishing  421
  list of objects
    example  423
  multiple table spaces
    example  423
  not writing changes to disk
    example  424
  table space set
    example  424
quiesce point, establishing  415
QUIESCE utility
  authorization  415
  catalog and directory objects  420
  compatibility  419
  creating point of consistency for catalog and directory  493
  data sets  418
  description  415
  example with lists  236
  examples  423
  failure to write to disk  422
  history record, printing  927
LISTDEF
  example  423
  lists  417
  options  416
  output  415
  partitions  417
  phases of execution  416
  prerequisite actions  418
  restarting  423
  restrictive states, compatibility  422
  syntax  416
  table space set  417
  terminating  423
  writing changed pages to disk  417

R
RBA (relative byte address), range printed by print log map  930
RBAEND
  option of DSN1LOGP  992
RBALRSN_CONVERSION
  option of LOAD  252
  option of REORG INDEX  532
  option of REORG TABLESPACE  601
RBASTART
  option of DSN1LOGP  992
RBDPEND
  option of REPAIR SET 684
RC0
  option of OPTIONS  410
RC4
  option of OPTIONS  410
RC8
  option of OPTIONS  410
  RCS, option of OPTIONS statement  410
RCPYDSN1
  option of DSNU CLIST  30
RCPYDSN2
  option of DSNU CLIST  30
REAL
  option of UNLOAD FROM TABLE  855
real-time statistics
  used by utilities  1123
REBALANCE
  option of REORG TABLESPACE  569
rebalancing partitions
  example  633
  rebinding, after LOAD  351
REBUILD
  option of REPAIR DBD  685
REBUILD INDEX utility
  access, specifying  445
  authorization  427
  building indexes in parallel  447
  catalog indexes  452
  compatibility  443
  data sets  440
  description  427
  DRAIN_WAIT, when to use  446
  dynamic sort and SORTDATA allocation, overriding  451
  examples  454
  FlashCopy  453
  index partitions  445
  options  431
  partition-by-growth table spaces  446
  performance recommendations  446
  phases of execution  427
  prerequisite actions  440
  recoverability of rebuilt index  452
  recycling version numbers  454
  restarting  453
  several indexes
    performance  446
    SHRLEVEL CHANGE
      when to use  446
  slow log processing  445
  sort subtasks for parallel build  450
  sort work file size  450
  syntax  428
  terminating  453
  work data sets, calculating size  441
REBUILD-pending (RBDP) status  452
  description  1043
  resetting  507, 1044
  set by LOAD utility  356
REBUILD-pending (RBDP*) status  452
REBUILD-pending status
  LOAD  346
rebuilding index
  clone tables  457
  example  454, 456, 457
  restrictive state  456
  SHRLEVEL CHANGE  456
rebuilding index partitions
  example 455
rebuilding indexes 447
RECD5
  option of DSNU CLIST 30
RECFM
  option of TEMPLATE 802
RECLUSTER
  option of REORG TABLESPACE 570
records
  order when loading 315
RECOVER utility
  authorization 459
backout
  point-in-time recovery 497
catalog and directory objects 485
CHECK-pending status, resetting 502
compatibility 473
compressed data, recovering 503
concurrent copies, improving recovery performance 466
damaged media, avoiding 509
data sets 472
description 459
DFSMShsm data sets 506
effects 510
error range 483
examples 511
fallback 508
FlashCopy 481
hierarchy of dependencies 490
incremental image copies 480
indexes
  REBUILD-pending status 511
JES3 environment 507
lists of objects 479
lists, using 463
LOB data 494
LOGAPPLY phase, optimizing 506
non-Db2 data sets 484
NOT LOGGED table spaces 484
objects accessed 490
options 463
output 459
pages, recovering 465, 483
parallel recovery 466
partitions, recovering 464, 480
performance recommendations 505
phases of execution 460
point-in-time recovery
  for catalog and directory objects 492
prerequisite actions 472
RBA, recovering to 465
rebalancing partitions with REORG 497
recovery status 499
restarting 509
restrictions 459
skipping copy or data set 504
syntax 461
tape mounts, retaining 509
terminating 509
XML data 494
RECOVER-pending (RECP) status
  description 1044
  resetting 356, 507, 1044
RECOVER-pending status
  LOAD 346
  recovery
  backout 497
catalog objects 485
clone objects 495
clone tables
  example 513
compressed data 503
consistency, ensuring 502
data set 480
database
  LOB table space 170
  RECOVER utility 459
directory objects 485
tape mounts, retaining 509
tape devices
  example 512
  to last image copy
  example 511
RECOVERY
  option of REPORT 705
recovery base 477
recovery index rows, deleting 394
recovery information
  reporting
    example 714
  reporting for index
    example 719
  reporting for partition
    example 717
    where it is stored 490
  recovery information, reporting 705
recovery log
  backward 913
  forward 912
recovery point of consistency
  creating for catalog and directory 493
recovery preparations
  QUIESCE 415
RECOVERY utility
  point-in-time recovery 495
RECOVERYDDN
  option of COPY 139
  option of COPYTOCOPY 191
  option of LOAD 287, 335
RECOVERYDDN (continued)
  option of MERGECOPY  380
  option of REORG TABLESPACE  572, 635
RECOVERYSITE
  option of RECOVER  471
  option of REPORT  708
referential constraint
  loading data  328
  violations  330
referential constraints
  LOAD
    example  367
REFRESH-pending (REFP) status  1045
RELOAD phase
  REORG TABLESPACE
    error  642
remote site recovery  155
REORG
  option of DSN1COMP  953
REORG INDEX utility
  access, allowing  521
  access, specifying  542
  authorization  515
  catalog updates  549
  CHECK-pending status, compatibility  535
  compatibility  539
  data sets  535
  data-sharing  534
  description  515
  drain behavior, specifying  524
  DRAIN_WAIT, when to use  546
  examples  549
  fallback recovery  534
  FlashCopy  544
  inline statistics
    gathering  528
    reporting  529
  interrupting  545
  lists, using  520
  long logs, actions for  524
  no action  527
  options  520
  output  515, 548
  partitions, specifying  521
  performance  545
  phases of execution  516
  preformatting pages  531
  prerequisite actions  534
  REBUILD-pending status, compatibility  534
  RECOVER-pending status, compatibility  534
  recycling version numbers  549
  region size  534
  report only  527
  restart-pending status, compatibility with SHRELEVEL
  CHANGE  534
  restarting  547
  retries, specifying maximum number  523
  shadow data sets  537
  SHRELEVEL CHANGE
  when to use  546
  SHRELEVEL option  542
  slow log processing, operator actions  543
  SWITCH phase deadline, specifying  522
  syntax  517
  terminating  546
  time for log processing, specifying  523
  timeout condition, actions for  525
REORG INDEX utility (continued)
  unload data sets
    specifying  531
  unloading data, action after  527
  waiting time when draining for SQL  523
REORG TABLESPACE utility
  access, specifying  573, 624
  actions after running  651
  authorization  554
  building indexes in parallel  638
  catalog and directory  603, 628
  compatibility  618
    with CHECK-pending status  606
    with REBUILD-pending status  605
    with RECOVER-pending status  605
    with REORG-pending status  606
  compression dictionary
    not building new  593
  data set
    copy, specifying  571
    discard, specifying  600
  data sets  609
  unload  612
    unload, specifying name  600
    work  613
  deadline for SWITCH phase, specifying  576
  description  553
  drain behavior, specifying  578
  DRAIN_WAIT, when to use  638
  dynamic sort work data set allocation, overriding  632
  effects  652
  encrypted data  606
  examples  654
  failed job, recovering  649
  fallback recovery  606
  FlashCopy  635
  indexes
    building in parallel  638
    inline copy  634
    inline statistics  594
    interrupting  632
    lists, using  565
    LOB table space  644
      phases of execution  556
      log processing, specifying max time  577
      logging, specifying  570
      long logs  578
      LONGLOG action, specifying interval  579
      mapping table  604
      specifying name  577
      multilevel security restrictions  555
      options  565
      output  553, 651
      parallel unloading and reloading  634
      partition-by-growth table spaces  642
      partitioned table spaces  642
      pending definition changes  646
      performance
        after adding column  395
        performance recommendations  637
      phases of execution  555
      preformatting pages  601
      prerequisite actions  603
      rebalancing  607
      rebalancing partitions  633
      reclaiming space from dropped tables  628
      records, discarding  602
REORG TABLESPACE utility (continued)
recycling version numbers 652
redistributing partitions 633
region size recommendation 603
RELOAD phase 644
RELOAD phase error 642
reload, skipping 628
restarting 649
sample generated LOAD statement 586
scope, specifying 568
segmented table spaces 644
selection condition 587
shadow data sets 614
SHRLEVEL
specifying 624
SHRLEVEL CHANGE
compatibility with restart-pending status 605
performance implications 637
when to use 637
slow processing, operator actions 627
sort device type, specifying 600
sort program messages, specifying destination 613
sort subtasks
allocation 641
determining number 640
sort work file, estimating size 641
syntax 556
temporary data sets, specifying number 600
terminating 647
time to wait for drain, specifying 577
timeout, specifying action 579
unload, specifying action 584
unloading data, methods of 642
XML table space 645
REORG utility
compressing data 331
REORG-pending (AREO*) status
description 1045
resetting 1046
REORG-pending (AREOR) status
description 1045
resetting 1046
REORG-pending (REORP) status
description 1045
resetting 1046
REORG-pending status
LOAD 346
reorganization
determining when to perform 542
reorganizing (continued)
list of indexes
example 550
parallel index build
example 655
partition
example 655
range of partitions
example 656
read-write access
example 655
REORG-pending partitions
example 666
reporting statistics
example 657
specifying maximum processing time
example 656
specifying unload data set
example 654
table space
example 654
table spaces, determining when to reorganize 622
updating statistics
example 657
REPAIR utility
actions after running 699
authorization 669
CATALOG statement 688
catalog, repairing 698
CHECK-pending status 699
compatibility 692
damaged page, repairing 695
data sets 691
DBD statement 684, 696
DELETE statement 680
description 669
DUMP statement 681
encrypted data 691
examples 699
LEVELID statement 686
LOCATE statement 675
logging, specifying 674
options 674
output 669, 698
output data sets
calculating size 691
partitions 686
phases of execution 670
prerequisite actions 690
REPLACE statement 679
resetting states, options 682
rows, locating by key 697
SET statement 682
syntax 670
system pages statement 689
terminating 698
VERIFY statement 678, 698
VERSIONS statement 687
REPLACE
option of LOAD 255
option of LOAD PART 283
statement of REPAIR utility 679
replacing data
LOAD
eample 363
replacing data in a table space 315
report option of REBUILD INDEX 435
option of REORG INDEX 529
option of REORG TABLESPACE 594
option of RUNSTATS INDEX 762
option of RUNSTATS TABLESPACE 756
REPORT utility
authorization 703
catalog and directory 709
compatibility 709
data sets 709
description 703
terminating 710
reporting
LOB columns
example 716
REPORTONLY
option of COPY 141, 168
option of REORG INDEX 527
option of REORG TABLESPACE 584
RESET
option of DSN1COPY 969
RESET ACCESSPATH
option of RUNSTATS TABLESPACE 758
resetting
auxiliary CHECK-pending (ACHKP) pending status 1037
auxiliary warning (AUW) warning status 1038
DBETE status 1041
pending status
CHECK-pending (CHKP) 1039
COPY-pending 1040
group buffer pool REBUILD-pending (GRECP) 1042
informational COPY-pending (ICOPY) 1042
page set REBUILD-pending (PSRBD) 1044
REBUILD-pending (RBDP) 507, 1044
RECOVER-pending (RECP) 507, 1044
REFRESH-pending (REFP) 1045
REORG-pending (AREO*) 1046
REORG-pending (AREOR) 1046
REORG-pending (REORP) 1046
reset-pending 1048
Persistent Read Only (PRO) restricted status 1043
REPORT
option of DSNJU003 914
REST
option of REPAIR REPLACE 679
restart
conditional control record
reading 940
RESTART
option of DSNU CLIST 30
restart-pending (RESTP) status
description 1048
resetting 1048
restarting (continued)
utilities 44
by using DSNUPROC 34
COPY 173
current restart 44
default value 46
LOAD 353
out-of-space condition 48
overriding default behavior 48
phase restart 44
REORG INDEX 547
REORG TABLESPACE 649
with lists 49
with templates 49
RESTORE SYSTEM utility
actions after running 738
authorization 731
compatibility 737
creating system point in time for 911
data sets 736
description 731
DISPLAY UTILITY command 737
effects of running 737
examples 738
options 733
output 731
phases of execution 732
prerequisite actions 734
REBUILD-pending status 737
resetting 737
syntax 732
terminating 737
RESTOREBEFORE
option of RECOVER 467
restricted status
after running LOAD 355
restrictive state 1037
restrictive states
resetting
example 700
restrictive status
resetting 695
RESUME
option of LOAD 253, 320
option of LOAD PART 283
RETAIN
option of MODIFY RECOVERY 391
RETPD
option of TEMPLATE 803
RETRY
option of CHECK DATA 79
option of CHECK INDEX 107
option of CHECK LOB 124
option of LOAD 275
option of REBUILD INDEX 434
option of REORG INDEX 523
option of REORG TABLESPACE 577
RETRY_DELAY
option of CHECK DATA 79
option of CHECK INDEX 107
option of CHECK LOB 124
option of LOAD 276
option of REBUILD INDEX 434
return code, CHANGELIMIT 168
return codes
altering 412
REUSE
  option of LOAD 264
  option of LOAD PART 253
  option of REBUILD INDEX 434
  option of RECOVER 466
  option of REORG INDEX 521
  option of REORG TABLESPACE 568
RI
  option of LISTDEF 223
RID
  option of DSN1LOGP 995
option of REPAIR LOCATE TABLESPACE 676
role
changing to 69
row change timestamp columns, loading 318
row format
  REORG TABLESPACE, effect of 653
ROWFORMAT
  option of LOAD 252
option of REORG TABLESPACE 601
ROWID
  option of LOAD 302
  option of REPAIR LOCATE TABLESPACE 677
  option of UNLOAD FROM TABLE 856
ROWID columns
loading 318, 347
ROWLIMIT
  option of DSN1COMP 953
RSTMBR
  option of DSNJU003 918
running
  utilities
    by creating JCL 36
RUNSTATS INDEX utility
  syntax 759
RUNSTATS utility
  access, specifying 756, 762
  after LOAD 351
  aggregation of statistics, specifying 758, 764
  authorization 742
  catalog table spaces 773
  catalog table updates 776
  COLGROUP option 741
  column frequency statistics, gathering 753
  column information, gathering 752
  compatibility 770
  data sets 767
  description 741
  device type for sort program, specifying 757, 763
  directory objects 773
  distribution statistics for column groups 771
  examples 783
  grouping columns 752
  HISTOGRAM option 741
  index frequency statistics, gathering 755
  INDEX option 741
  index partitions, specifying 760
  INDEX syntax diagram 759
  key column combinations, gathering information 754
  lists, using 749, 760
  LOB table space, space statistics 774
  options for RUNSTATS INDEX 760
  output 742, 776
  partitioned table space 772
  phases of execution 743
  prerequisites 741
  profile syntax 764
RUNSTATS utility (continued)
  reporting information 756, 762
  restarting 743
  sample of columns, gathering statistics 750
  sort work data sets, specifying number 757, 763
  statistics history 774
  table space partitions, gathering statistics 749
  TABLESPACE option 741
  TABLESPACE options 749
  TABLESPACE syntax 743
  terminating 743
  updating catalog information 756, 762
  work data sets
    frequency statistics 773
    XML table space, collect statistics 774
RUNSTATS utility
  resetting access path statistics 780
S
SAMPLE
  option of REORG TABLESPACE 594
  option of RUNSTATS TABLESPACE 750
  option of UNLOAD FROM TABLE 844
schema
renaming 69
SCHEMA SWITCH
  option of CATMAINT 66
SCOPE
  option of CHECK DATA 79, 94
  option of COPY 147
  option of REBUILD INDEX 434
  option of REORG TABLESPACE 568
SECPORT
  option of DSNJU003 914
security
  data sets 18
  utilities 5
SEGMENT
  option of DSN1COPY 966
segmented table spaces
  reorganizing 644
SELECT
  option of DSN1SDMP 1024
SELECT2
  option of DSN1SDMP 1027
service level
  finding
    DIAGNOSE example 208
SET
  statement of REPAIR utility 682
SET PROFILE
  option of RUNSTATS TABLESPACE 752
SETCURRENTVERSION
  option of REPAIR system pages 690
shadow data sets
  CHECK DATA 88
  CHECK INDEX 109
  CHECK LOB 126
  REORG INDEX 537
  REORG TABLESPACE 614
shortcut keys
  keyboard xvii
SHRLEVEL
  option of CHECK DATA 78
  option of CHECK INDEX 106
  option of CHECK LOB 123
SHRLEVEL (continued)
  option of COPY 146, 162
  option of LOAD 254
  option of REBUILD INDEX 432, 445
  option of REORG INDEX 521
  option of REORG TABLESPACE 573
  option of REPAIR LOCATE TABLESPACE 677
  option of RUNSTATS INDEX 762
  option of RUNSTATS TABLESPACE 756
  option of UNLOAD 835
SHRLEVEL CHANGE
  CHECK DATA, example 102
  option of REPAIR system pages 690
SIZE
  option of DSNUPROC 34
SKIP
  option of OPTIONS 409
  skipping errors
    OPTIONS example 413
SMALLINT
  option of LOAD 296
  option of UNLOAD FROM TABLE 851
sort program
  data sets for REORG TABLESPACE, specifying device type 600
  messages from REORG TABLESPACE, specifying destination 613
SORTCLUSTER
  option of REORG TABLESPACE 569
SORTDATA
  option of REORG TABLESPACE 570
SORTDEVT
  option of CHECK DATA 84
  option of CHECK INDEX 107
  option of CHECK LOB 124
  option of LOAD 272
  option of REBUILD INDEX 435
  option of REORG INDEX 531
  option of REORG TABLESPACE 600
  option of RUNSTATS INDEX 763
  option of RUNSTATS TABLESPACE 757
SORTKEYS
  option of LOAD 265
SORTNPSI
  option of REORG TABLESPACE 581
SORTNUM
  option of CHECK DATA 85
  option of CHECK INDEX 107
  option of CHECK LOB 125
  option of LOAD 272
  option of REBUILD INDEX 435
  option of REORG INDEX 531
  option of REORG TABLESPACE 600
  option of RUNSTATS INDEX 763
  option of RUNSTATS TABLESPACE 757
SORTOUT
  LOAD utility, estimating size 309
  space
    DBD, reclaiming 395
    unused, finding for nonsegmented table space 623
SPACE
  option of MODIFY STATISTICS 401
  option of TEMPLATE 806
SPACE column
  analyzing values 791
SPACEF column
  analyzing values 791
SPANNED
  option of LOAD 269
  option of UNLOAD 832
spanned record format
  unloading in 866
SQL statement records 1089
STACK
  option of TEMPLATE 808
stand-alone utilities 891
  control statements 893
definition 1
  invoking 891
  JCL EXEC PARM, using to specify options 892
  specifying options 892
standard report header
  CONNECT field 1056
  CONNTYPE field 1056
  CORRNAME field 1056
  CORRNMBR field 1056
  DESCRIPTION field 1057
  DESTNO field 1057
  END_USER field 1056
  IFCID field 1057
  INSTANCE field 1056
  ORIGAUTH field 1056
  PLANNENAME field 1056
  PRIMAUTH field 1056
  RECORD TIME field 1056
  TCB CPU TIME field 1057
  TRANSACT field 1057
  WS_NAME field 1057
START
  option of CATENFM 62
START TRACE
  option of DSN1SDMP 1023
STARTTIME
  option of DSNJU003 910
STARTRBA
  option of DSNJU003 908
STATCLGMEMSR
  option of RUNSTATS TABLESPACE 755
statistics
  access path, updating
    example 784
  data-partitioned secondary indexes 773
  deleting access path records
    example 404
  deleting for index space
    example 405
  deleting history records
    example 404
  deleting space-tuning records
    example 404
  gathering 741
  generating report only
    example 784
  histogram
    example 786
  history 774
  prohibiting updates while gathering
    example 783
  retrieving least frequently occurring values
    example 785
  retrieving most frequently occurring values
    example 785
  updating
    example 783
Statistics (continued)
  updating and generating report  784
  updating catalog and history tables  784
  updating for a column group  785
  updating for column group  785
  updating for partition  784
  updating for several tables  783
  updating for table space  783
  updating frequently occurring values  784
Statistics
  access paths
    resetting  780
STASTISTICS
  option of LOAD  258
  option of REBUILD INDEX  435
  option of REORG INDEX  528
  option of REORG TABLESPACE  594
statistics history
  deleting specific entries  403
  reasons to delete  403
status
  CHECK-pending
    resetting  357
  COPY-pending
    resetting  356
  displaying  1037
  page set REBUILD-pending (PSRBD)  452
  REBUILD-pending (RBDF)  452
  REBUILD-pending star (RBDP*)  452
  utilities  39
  status of utility
    active  39
    stopped  39
    terminated  39
STOGROUP
  option of STOSPACE  788
  stopped
    utility status  39
  storage group
    monitoring disk space  789
  storage group name
    with spaces
      STOSPACE example  791
STORCLAS
  option of TEMPLATE  802
stored procedures
  Db2-supplied  1035
STOSPACE utility
  authorization  787
  availability of objects, ensuring  789
  compatibility  789
  data sets  788
  description  787
  examples  791
  monitoring disk space for a storage group  789
  options  788
  output  787
  phases of execution  787
  restarting  791
STOSPACE utility (continued)
  statistical information, obtaining  789
  syntax  788
  terminating  791
string
  naming convention  xix
STRIP
  option of LOAD  342
  option of UNLOAD FROM TABLE VARCHAR  848
  option of UNLOAD FROM TABLE VARCHAR  848
  option of UNLOAD FROM TABLE VARGRAPHIC  850
  VARCHAR  342
STRTLRSN
  option of DSNJU003  909
STTRACE
  option of DSN1SDMP  1025
SUBMIT
  option of DSN1SDMP  1025
substring notation, TEMPLATE utility  799
SUBSYS
  option of TEMPLATE  801
  subsystem
    backing up  51
    collecting information  1099
    naming convention  xix
    restoring  731
  subsystem parameters
    DFSMSsды COPY operation with utilities  40
  subsystem, copying tables from one to another  981
SUBTYPE
  option of DSN1LOGP  996
SUMMARY
  option of DSN1LOGP  997
  option of REPORT  708
switching
  templates  815
SWITCHTIME
  option of LOAD  276
  option of REORG INDEX  525
  option of REORG TABLESPACE  580
syntax
  BACKUP SYSTEM  52
  CATENFM  61
  CATMAINT  66
  CHECK DATA  74
  CHECK INDEX  104
  CHECK LOB  122
  COPY  135
  COPYTOCOPY  186
  DIAGNOSE  204
  DSNNCOMP  951
  DSNNCOPY  965
  DSNNLOGP  991
  DSN1PRNT  1011
  DSNN1SDMP  1023
  DSNNADMSB  1100
  DSNJU003  904
  DSNJU004 (print log map)  929
  DSNJU008 (print CDDS utility)  946
  DSNU CLIST command  28
  EXEC SQL  211
  LISTDEF  215
  LOAD  242
  MERGECOPY utility  378
  MODIFY RECOVERY  389
  MODIFY STATISTICS  400
  OPTIONS  407
syntax (continued)
QUIESCE utility 416
REBUILD INDEX 428
RECOVER utility 461
REORG INDEX utility 517
REORG TABLESPACE utility 556
REPAIR 670
REPORT 704
RESTORE SYSTEM 732
RUNSTATS INDEX 759
RUNSTATS profile 764
RUNSTATS TABLESPACE 743
STOSP ACE 788
TEMPLATE 793
UNLOAD 825

SYSCOPY
deleting clone records
example 398
deleting records example 397
deleting records by age
example 397
deleting records by date
example 397
deleting records for partitions
example 397
option of DSN1LOGP 993
removing outdated information 387
retaining GDG records
MODIFY RECOVERY example 398
retaining records
MODIFY RECOVERY example 398
SYSCOPY, deleting rows 393
SYSDBDXA directory table space
MERGECOPY restrictions 378, 379
SYSDIC data set
LOAD utility, estimating size 310
SYSSERR data set
LOAD utility, estimating size 310
SYSIBM.SYSCOPY
copying 153
COPYT OCOPY 196
ICBACKUP column 155
ICUNIT column 155
SYSIBM.SYSCOPY catalog table
FlashCopy image copies 160
SYSIBM.SYSLGRNX
copying 153
SYSIBM.SYSUTILX
copying 153
SYSLGRNX
deleting clone records
example 398
deleting records by age
example 397
deleting records by date
example 397
removing outdated information 387
retaining GDG records
MODIFY RECOVERY example 398
retaining records
MODIFY RECOVERY example 398
SYSLGRNX, deleting rows 393
SYSLGRNX data set
LOAD utility, estimating size 310
SYSLGRNX data set
LOAD utility, estimating size 310
SYSMAP data set
LOAD utility, estimating size 310
SYSPITR
option of DSNJU003 911
SYSPITRT
option of DSNJU003 912
SYSPRINT DD
DSN1SMFP 1097
SYSTEM
option of DSNJU008 947
option of DSNU CLIST 31
option of DSNUPROC 34
system data sets
renaming 924
system pages
statement of REPAIR utility 689
system parameter records 1066
system point in time, creating 911
system recovery
specifying a point in time
DSNJU003 example 926
system-level backups
determining which ones are used for recovery 476
DSNJU004 output 942
recovering with 475
system-level-backups
determining whether on disk or tape 477
system-period temporal table spaces
reporting
example 720
SYSTEMPAGES
option of COPY 143
SYSTSCPY directory table space
MERGECOPY restrictions 378, 379
SYSUT1 data set
LOAD utility, estimating size 310
SYSTC directory table space
MERGECOPY restrictions 378, 379
order of recovering 487

T

Table
adding data 320
dropping, reclaiming space 628
exception, creating 92
multiple, loading 277
replacing data 315
retrieving data about
DSNADMSB example 1113
TABLE
option of LISTDEF 221
option of REORG TABLESPACE 594
option of RUNSTATS TABLESPACE 749
table name
naming convention xix
table space
checking 73
determining when to reorganize 542, 622
merging copies 377
naming convention xx
nonsegmented, finding unused space 623
recovering 478
status, resetting 695
table space set
reporting
example 720
TABLESAMPLE SYSTEM
  option of RUNSTATS TABLESPACE 751
TABLESPACE
  option of CHECK DATA  76
  option of CHECK INDEX 106
  option of CHECK LOB 123
  option of COPY 138
  option of COPYTOCOPY 188
  option of LISTDEF  220
  option of MERGECOPY 379
  option of MODIFY RECOVERY 389
  option of MODIFY STATISTICS 400
  option of QUIESCE  417
  option of REBUILD INDEX 432
  option of RECOVER  464
  option of REPAIR  674
  option of REPAIR CATALOG 689
  option of REPAIR LEVELID 686
  option of REPAIR SET  682
  option of REPAIR system pages 690
  option of REPAIR VERSIONS 687
  option of REPORT  706
  option of RUNSTATS INDEX 760
  option of UNLOAD  828
TABLESPACES
  option of LISTDEF  217
TABLESPACESET
  option of QUIESCE  417
  option of REPORT  708
TAPEUNITS
  option of COPY  143
  option of RECOVER 467
  option of RESTORE SYSTEM 733
template
  for disk
    example  817
  for image copy
    example  816
  for tape data sets
    example  816
  for unloading LOB objects
    example  819
GDG data set
  example  817, 818
switching
  example  819
  variable substring notation
    example  816
  with LISTDEF
    example  817
TEMPLATE library  810
  example  413
TEMPLATE library, specifying  411
template switching
  COPY
    example  177
  COPYTOCOPY
    example  201
  example  385
TEMPLATE utility (continued)
data set size
  default space calculations  812
  extent allocations  813
description  793
devices
  specifying number  803
disposition of data set
  specifying  803
examples  816
expiration date for data set, specifying  803
GDG base, specifying number of entries  803
GDGs  815
model data set, specifying  802
operations  811
options  797
output  793
phases of execution  793
prerequisite actions  810
PREVIEW mode, executing in  811
previewing data set names  812
restarting  815
retention period for data set, specifying  803
scope of control statement  811
SMS data class, specifying  802
SMS management class, specifying  802
SMS storage class, specifying  802
space parameters, specifying  806
substring notation  799
switching  815
syntax  793
tape  813
terminating  815
track recording technique, specifying  808
variables
  DATE  801
  TIME  801
  using in the data set name  798
volumes, specifying maximum number  803
volumes, specifying

TERMINAL Utility command
BACKUP SYSTEM  57
CATENFM  64
COPY  172
COPYTOCOPY  198
description  43
LISTDEF  234
LOAD  352
MERGECOPY  384
MODIFY STATISTICS 403
OPTIONS  412
QUIESCE  423
REBUILD INDEX 453
RECOVER  509
REORG INDEX  546
REORG TABLESPACE  647
REPAIR  698
REPORT  710
STOSPACES  791
TEMPLATE  815
terminated
  utility status  39
terminating
  online utilities  43
terminating (continued) utilities

BACKUP SYSTEM 57
CATENFM 64
CATMAINT 68
CHECK DATA 99
CHECK INDEX 118
CHECK LOB 130
COPY 172
COPYTOCOPY 198
data sharing 43
DIAGNOSE 208
EXEC SQL 213
LISTDEF 234
LOAD 352
MERGECOPY 384
MODIFY STATISTICS 403
OPTIONS 412
QUIESCE 423
REBUILD INDEX 453
RECOVER 509
REORG INDEX 546
REORG TABLESPACE 647
REPAIR 698
REPORT 710
RESTORE SYSTEM 737
STOSPICE 791
TEMPLATE 815
UNLOAD 825
TEST
option of REPAIR CATALOG 689
option of REPAIR DBD 685
TIME
option of DSNJU003 917
option of TEMPLATE 806
TIME EXTERNAL
option of LOAD 300
option of UNLOAD FROM TABLE 855
TIMEOUT
option of REORG INDEX 525
option of REORG TABLESPACE 579
TIMESTAMP EXTERNAL
option of LOAD 301
option of UNLOAD FROM TABLE 855
TIMESTAMP WITH TIME ZONE EXTERNAL
option of LOAD 302
TIMESTAMP WITH TIMEZONE EXTERNAL
option of UNLOAD FROM TABLE 855
timestamp, BSDS
DSNJU004 output 937
timestamps, printing system and utility 927
TOCOPY
option of RECOVER 468
TOKEN
option of BACKUP SYSTEM 54
TOLASTCOPY
option of RECOVER 469
TOLASTFULLCOPY
option of RECOVER 469
TOLOGPOINT
option of RECOVER 465
TORMA
option of RECOVER 465
TOSEQNO
option of RECOVER 469
TOVOLUME
option of RECOVER 469
trace records 1051
decompressing
DSNTSMFD 1119
element 1120
trace start and stop with AUDITPOLICY records 1095
trace start records 1059
trace stop records 1059
TRACEID
option of DIAGNOSE 207, 210
traces
assigning to buffers 1027
tracing
processor use by utilities 40
TRTCH
option of TEMPLATE 808
TRUNCATE
option of LOAD 342
VARCHAR 342
trusted connection records 1087
trusted context
utilities 5
trusted context records 1089
TYPE
option of DIAGNOSE 205
option of DSN1LOGP 996

U
UID
option of DSNU CLIST 31
option of DSNUPROC 34
UNCNT
option of TEMPLATE 803
UNICODE
option of LOAD 270
option of UNLOAD 831
Unicode data
LOAD
example 372
UNIT
option of DSNJU003 909
option of DSNU CLIST 31
option of TEMPLATE 802
UNLDDN
option of REORG TABLESPACE 600
option of UNLOAD 830
UNLOAD
option of REORG INDEX 527
option of REORG TABLESPACE 584
UNLOAD utility
access, specifying 835
ASCII format, specifying 831
authorization 821
blanks in VARBINARY fields, removing 852
blanks in VARCHAR fields, removing 848
blanks in VARCHARGRAPHIC fields, removing 850
BLOB data type, specifying 856
BLOB strings, truncating 857
CCSID, specifying 831
CHAR data type, specifying 846
character string representation of date, specifying 855
character string representation of time, specifying 855
character strings, truncating 847
CLOB data type, specifying 857
CLOB strings, truncating 857
compatibility 823
compressed data 881
UNLOAD utility (continued)
constant field, specifying 835
converting data types 870
CURRENT DATE, incrementing and decrementing value 863
data sets 822
data type compatibility 871
DBCLOB format, specifying 857
DBCS string, truncating 858
DD statement for image copy, specifying 829
decimal format, specifying 852
delimited files 877
delimited format
  decimal point character 835
delimited format, specifying 834
delimiters
  column 834
  string 834
description 821
EBCDIC format, specifying 831
encrypted data
  running UNLOAD on 822
examples 881
field position, specifying 845
field specification errors 881
field specifications 838
floating-point data, specifying format 854
FROM TABLE clause 838, 867
  compatibility with LIST 838
FROM TABLE options 843
FROM TABLE syntax diagram 839
graphic type, specifying 849
graphic type, truncating 850
header field, specifying 843
image copies
  concatenating 869
  unloading from 868
image copy, specifying 828
Infinity 876
integer format, specifying 851
internal format, specifying 833
lists, specifying 829
LOAD statements, generating 880
LOAD statements, specifying data set for 829
LOB data 865
maximum errors allowed, specifying 835
maximum number of rows to unload, specifying 845
multiple table spaces, unloading 838
NaN 876
options 827
output 821
output columns
  ordering 868
output fields
  layout 873
  position and size 873
  types 871
padding for variable length data, not using 832
partitions 864
partitions, identifying 828
phases of execution 822
prerequisites 822
processing encrypted data 822
restarting 825
restrictions 822
ROWID type, specifying for output data 856
UNLOAD utility (continued)
sampling rows 844
selection condition 858
small integer, specifying 851
sNaN 876
source tables, selecting 867
spanned record format 866
substitutions, not using 832
syntax 825
table space, specifying 828
terminating 825
timestamp, incrementing and decrementing value 864
truncating data 879
Unicode format, specifying 831
unload data set, specifying 830
varying-length data format, specifying 847
varying-length graphic type, specifying 850
WHEN clause 858
XML data 864
unloading
  converting data
    example 886
  field specification list
    example 882
  from clone tables
    example 887
  from image copy
    example 882
  from partitioned table space in parallel
    example 883
  from two tables
    example 883
  LOB data
    example 886
  methods 642
  multiple table spaces
    example 884
  sample of rows
    example 883
  segmented table space
    example 883
  to delimited file
    example 885
  Unicode data to EBCDIC
    example 886
  with lists
    example 884
UPDATE
  option of CATMAINT 66
  option of REBUILD INDEX 435
  option of REORG INDEX 529
  option of REORG TABLESPACE 594
  option of RUNSTATS INDEX 762
  option of RUNSTATS TABLESPACE 756
UPDATE PROFILE
  option of RUNSTATS TABLESPACE 752
URID (unit of recovery ID)
  option of DSN1LOGP 995
USE PROFILE
  option of REORG TABLESPACE 594
  option of RUNSTATS TABLESPACE 750
utilities
  authorization IDs 5
  controlling 39
  declared temporary tables 7
  executing
    by using DSNUPROC 33
utilities (continued)
executing (continued)
   DB2I 22
   JCL 36
invoking
   DSNU CLIST command 26
mixed-release data sharing environment, operating in
   monitoring 39
online
   BACKUP SYSTEM 51
   CATENFM 61
   CATMAINT 65
   CHECK DATA 73
   CHECK INDEX 103
   CHECK LOB 121
   COPY 133
   COPYTOCOPY 185
   DIAGNOSE 203
   EXEC SQL 21
   invoking 15
   LISTDEF 215
   LOAD 239
   MERGECOPY 377
   MODIFY RECOVERY 387
   MODIFY STATISTICS 399
   OPTIONS 407
   QUIESCE 415
   REBUILD INDEX 427
   RECOVER 459
   REORG INDEX 515
   REORG TABLESPACE 553
   REPAIR 669
   REPORT 703
   RESTORE SYSTEM 731
   RUNSTATS 741
   STOPSPACE 787
   TEMPLATE 793
   UNLOAD 821
overview 1
packaging 3
phase, determining 39
privileges 5
real-time statistics 1123
restarting 44
running in preview mode 411
stand-alone 891
   DSNICOMP 949
   DSNICOPY 961
   DSNLOGP 989
   DSNIPRNT 1009
   DSNISDMP 1021
   DSNJCNVB 895
   DSNJCNVT 897
   DSNJLOGF (preformat active log) 901
   DSNJU003 (change log inventory) 903
   DSNJU004 (print log map) 927
   DSNJU008 (print CDDS) 945
invoking 891
traded context 5
UTILITIES panel 23
UTILITY
   option of DSNU CLIST 29
utility abend
   forcing
      DIAGNOSE example 209
utility change records 1063
utility control statements
   overview 18
   parsing rules 18
   stand-alone utilities 893
utility end records 1064
utility execution
   suspending
      DIAGNOSE example 209
utility failure
   determining reason 42
utility start records 1060
utility-id
   naming convention  xx
UTILX
   option of CATMAINT 67
UTPROC
   option of DSNUPROC 34

V
validation routine
   LOAD 239
VALUE
   option of DSN1COPY 969
   option of DSN1LOGP 997
   option of DSN1PRNT 1015
VARBINARY
   option of LOAD 297
   option of UNLOAD FROM TABLE 852
VARCHAR
   data type, loading 315
   option of LOAD 292
   option of UNLOAD FROM TABLE 847
VARGRAPHIC
   data type, loading 315
   option of LOAD 295
   option of UNLOAD FROM TABLE 850
varying-length rows, relocated to other pages, finding number of 623
VCAT SWITCH
   option of CATMAINT 67
VER
   option of DSNJU008 947
VERIFY
   statement of REPAIR utility 678
VERIFYSET
   option of RECOVER 470
VERSION
   option of REPAIR LOCATE TABLESPACE 677
version information
   updating
      example 700
   updating on the same system 687
version numbers
   recycling
      LOAD 360
      MODIFY RECOVERY 396
      REBUILD INDEX 454
      REORG INDEX 549
      REORG TABLESPACE 652
versioning relationships
   reporting
      example 720
VERSIONS
   statement of REPAIR utility 687
violations
   correct 95

Index 1171
violations (continued)
finding  94

VOLCNT
  option of TEMPLAT E  803

VOLUME
  option of DSNU CLIST  31

VOLUMES
  option of TEMPLAT E  803

VSAM (Virtual Storage Access Method)
  used by STOSPACE  789

VSAMCAT
  option of DSNJU003  913

W

WAIT
  option of DIAGNOSE  206

WARNING
  option of OPTIONS  410

WHEN
  option of LOAD  285
  option of REORG TABLESPACE  587
  option of UNLOAD FROM TABLE  858

work data sets
  CHECK DATA  84
  CHECK DATA utility  87
  LOAD utility  309

WORKDDN
  option of CHECK DATA  84
  option of LOAD  265
  option of MERGECOPY  380
  option of REORG INDEX  531

WRITE
  option of QUIESCE  418

X

XML
  option of LISTDEF  224
  option of LOAD  303
  option of UNLOAD FROM TABLE  858

XML column
  loading  349

XML columns
  example of checking  102
  reporting
    example  720

XML data
  collecting statistics  774
  correcting after CHECK INDEX  118
  loading  324
  recovering  494
  unloading  864
  unloading to spanned record format  866

XML schema repository
  copying  165

XML table space
  copying  164, 185
  LOAD LOG  350
  reorganizing  645

XMLCHECK DATA  95

XMLERROR
  option of CHECK DATA  81